Lecture Notes in Artificial Intelligence 2464

Subseries of Lecture Notes in Computer Science
Edited by J. G. Carbonell and J. Siekmann

Lecture Notes in Computer Science
Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

VISIT…

Springer
Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Tokyo

Michael O'Neill Richard F. E. Sutcliffe
Conor Ryan Malachy Eaton
Niall J. L. Griffith (Eds.)

Artificial Intelligence and Cognitive Science

13th Irish Conference, AICS 2002
Limerick, Ireland, September 12-13, 2002
Proceedings

Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Volume Editors

Michael O'Neill
Richard F. E. Sutcliffe
Conor Ryan
Malachy Eaton
Niall J. L. Griffith
University of Limerick
Department of Computer Science and Information Systems
Ireland
{michael.oneill, richard.sutcliffe, conor.ryan, malachy.eaton, niall.griffith}@ul.ie

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Artificial intelligence and cognitive science : 13th Irish conference ;
proceedings / AICS 2002, Limerick, Ireland, September 12 - 13, 2002.
Michael O'Neill ... (ed.). - Berlin ; Heidelberg ; New York ; Barcelona ;
Hong Kong ; London ; Milan ; Paris ; Tokyo : Springer, 2002
(Lecture notes in computer science ; 2464 : Lecture notes in artificial
intelligence)
ISBN 3-540-44184-0

CR Subject Classification (1998): I.2, F.1

ISSN 0302-9743
ISBN 3-540-44184-0 Springer-Verlag Berlin Heidelberg New York

Springer-Verlag Berlin Heidelberg New York,
a member of BertelsmannSpringer Science+Business Media GmbH

http://www.springer.de

Typesetting: Camera-ready by author, data conversion by PTP-Berlin, Stefan Sossna e.K.
Printed on acid-free paper SPIN: 10871241 06/3142 5 4 3 2 1 0

Preface

The Artificial Intelligence and Cognitive Science Conference has taken place annually since 1988. It provides a forum for the exchange of ideas and the presentation of results relating to work conducted both in Ireland and worldwide.

The conference spans a large number of fields including case-based reasoning, cognitive modeling, constraint processing, data mining, evolutionary computation, intelligent agents, intelligent information retrieval, knowledge representation and reasoning, learning, natural language processing, neural networks, perception and planning, robotics, and scheduling.

AICS 2002 was the thirteenth conference in the series and took place at University of Limerick on 12–13 September. In addition to the 16 regular papers and 17 concise papers accepted for presentation, we were delighted to welcome two prestigious keynote speakers both fitting in with the conference theme 'Towards Bio-inspired Computing'. They were David Goldberg of University of Illinois at Urbana-Champaign, and Dario Floreano of the Swiss Federal Institute of Technology.

I would like to take this opportunity to thank our sponsors QAD Ireland and University of Limerick, as well as all those who were involved in the organisation of the conference including the Co-chairs, Programme Committee members, and Conference Administrators.

June 2002 Richard F. E. Sutcliffe

Organisation

AICS 2002 was organized by the Department of Computer Science and Information Systems, University of Limerick, in association with the Artificial Intelligence Association of Ireland.

Organising Committee

General Chair:	Richard F.E. Sutcliffe
Publication Chair:	Michael O'Neill
Program Co-chair:	Conor Ryan
Local Arrangements Chair:	Malachy Eaton
Program Co-chair:	Niall Griffth
Conference Administrators:	Maeve Gleeson, Bianca Pluk

Programme Committee

Liam Bannon, University of Limerick
David Bell, University of Ulster at Jordanstown
Anthony Brabazon, University College Dublin
Derek Bridge, University College Cork
Jim Buckley, University of Limerick
Ruth Byrne, The University of Dublin, Trinity College
Arthur Cater, University College Dublin
J.J. Collins, University of Limerick
Ernesto Costa, University of Coimbra
Fintan Costello, Dublin City University
Roddy Cowie, Queen's University Belfast
Norman Creaney, University of Ulster at Coleraine
Fred Cummins, University College Dublin
Pádraig Cunningham, The University of Dublin, Trinity College
John Dunnion, University College Dublin
Jim Eales, University of Limerick
Malachy Eaton, University of Limerick
José Luis Fernández-Villacañas, Universidad Carlos III de Madrid
Dario Floreano, Swiss Federal Institute of Technology
Antony Galton, University of Exeter
Paul Gibson, National University of Ireland, Maynooth
David Goldberg, University of Illinois at Urbana-Champaign
Josephine Griffith, National University of Ireland, Galway
Niall Griffith, University of Limerick
Jane Grimson, The University of Dublin, Trinity College

Patrick Healy, University of Limerick
Michael Herdy, INPRO
Gareth Jones, University of Exeter
Mark Keane, University College Dublin
Maarten Keijzer, Vrije University, Amsterdam
Nick Kushmerick, University College Dublin
Michael Luck, University of Southampton
Annette McElligott, University of Limerick
Paul Mc Kevitt, University of Ulster at Magee
David McSherry, University of Ulster at Coleraine
Mike McTear, University of Ulster at Jordanstown
Juan Julián Merelo, University of Granada
Liam Murray, University of Limerick
Fionn Murtagh, Queen's University Belfast
Ajit Narayanan, University of Exeter
Diarmuid O'Donoghue, National University of Ireland, Maynooth
Greg O'Hare, University College Dublin
Donncha Ó Maidín, University of Limerick
Michael O'Neill, University of Limerick
Colm O'Riordan, National University of Ireland, Galway
Riccardo Poli, University of Essex
Ronan Reilly, National University of Ireland, Maynooth
Conor Ryan, University of Limerick
Reinhard Schaeler, University of Limerick
Jonathan Shapiro, University of Manchester
Noel Sharkey, University of Sheffield
Barry Smyth, University College Dublin
Humphrey Sorensen, University College Cork
Richard F. E. Sutcliffe, University of Limerick
Adrian Trenaman, Iona Technologies
Josef van Genabith, Dublin City University
Carl Vogel, The University of Dublin, Trinity College
Andy Way, Dublin City University
Adam Winstanley, National University of Ireland, Maynooth

Sponsors

QAD Ireland
University of Limerick

Table of Contents

Regular Papers

Concise Papers

On the Usefulness of Extracting Syntactic Dependencies for Text Indexing*

Miguel A. Alonso[1], Jesús Vilares[1,2], and Víctor M. Darriba[2]

[1] Departamento de Computación, Universidade da Coruña
Campus de Elviña s/n, 15071 La Coruña, Spain
alonso@udc.es, jvilares@mail2.udc.es
http://coleweb.dc.fi.udc.es/
[2] Escuela Superior de Ingeniería Informática, Universidade de Vigo
Campus de As Lagoas, 32004 Orense, Spain
jvilares@uvigo.es, darriba@ei.uvigo.es

Abstract. In recent years, there has been a considerable amount of interest in using Natural Language Processing in Information Retrieval research, with specific implementations varying from the word-level morphological analysis to syntactic parsing to conceptual-level semantic analysis. In particular, different degrees of phrase-level syntactic information have been incorporated in information retrieval systems working on English or Germanic languages such as Dutch. In this paper we study the impact of using such information, in the form of syntactic dependency pairs, in the performance of a text retrieval system for a Romance language, Spanish.

1 Introduction

For Information Retrieval (IR) tasks, documents are frequently represented through a set of index terms or representative keywords. This can be accomplished through operations such as the elimination of *stopwords* (too frequent words or words with no apparent significance) or the use of *stemming* (which reduces distinct words to their supposed grammatical root). These operations are called *text operations*, providing a *logical view* of the processed document. More elaborated index terms can be created by combining two or more content words (nouns, verbs and adjectives) in a *multi-word term* [11,6,12]. Most techniques for extracting multi-word terms rely on statistics [7] or simple pattern matching [12], instead of considering the structural relations among the words that form a sentence. In this paper, we propose to use practical, finite-state, Natural Language Processing (NLP) techniques to extract such multi-word terms in the form of pairs of words related by some kind of syntactic dependency.

* The research reported in this article has been supported in part by Plan Nacional de Investigación Científica, Desarrollo e Innovación Tecnológica (Grant TIC2000-0370-C02-01), Ministerio de Ciencia y Tecnología (Grant HP2001-0044) and Xunta de Galicia (Grant PGIDT01PXI10506PN).

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 3–11, 2002.

2 Extraction of Syntactic Dependencies

Given a stream of tagged words, we want to obtain the *head-modifier* pairs corresponding to the most relevant syntactic dependencies [5]: *Noun-Modifier*, relating the head of a noun phrase with the head of a modifier; *Subject-Verb*, relating the head of the subject with the main verb of the clause; and *Verb-Complement*, relating the main verb of the clause with the head of a complement. It has to be noted that while the head-modifier relation may suggest semantic dependence, what we obtain here is strictly syntactic, even though the semantic relation is what we are really after [16].

The kernel of the grammar used by our shallow parser has been inferred from the basic trees corresponding to noun phrases and their syntactic and morpho-syntactic variants [11]. *Syntactic variants* result from the inflection of individual words and from modifying the syntactic structure of the original noun phrase. *Morpho-syntactic variants* differ from syntactic variants in that at least one of the content words of the original noun phrase is transformed into another word derived from the same morphological stem. At this point we must recall that inflectional morphemes represent grammatical concepts such as gender, person, mood, or time and tense. On the other hand, derivational morphemes effect a semantic change on the base, often also effecting a change of syntactic class. We define a *morphological family* as the set of words obtained from the same morphological root through derivation mechanisms, such as prefixation, emotive suffixation, non-emotive suffixation, back formation and parasynthesis. A system for the automatic generation of morphological families has been described in [18].

2.1 Syntactic Variants

The example of Fig. 1 shows the basic structure of a noun phrase and some of its possible syntactic variants, together with the syntactic dependencies they contain. Such variants were obtained by applying to the source phrase *una caída de las ventas* (a drop in the sales) the following mechanisms [11]:

- *Synapsy:* a unary construction which corresponds to a change of preposition or the addition or removal of a determiner.
- *Substitution:* it consists of employing modifiers to make a term more specific.
- *Permutation:* this refers to the permutation of words around a pivot element.
- *Coordination:* this consists of employing coordinating constructions (copulative or disjunctive) with the modifier or with the modified term.

Symbols A, C, D, N, P, V and W are the part-of-speech labels that denote adjectives, coordinating conjunctions, determiners, nouns, prepositions, verbs and adverbs, respectively. In addition, we have conflated each word in a dependency pair by replacing it with an identifier of its morphological family (actually, one of the words in such family, its *representative*).

The structures and syntactic dependencies corresponding to all the syntactic variants shown in Fig. 1 are embedded in the syntactic pattern shown in Fig. 2

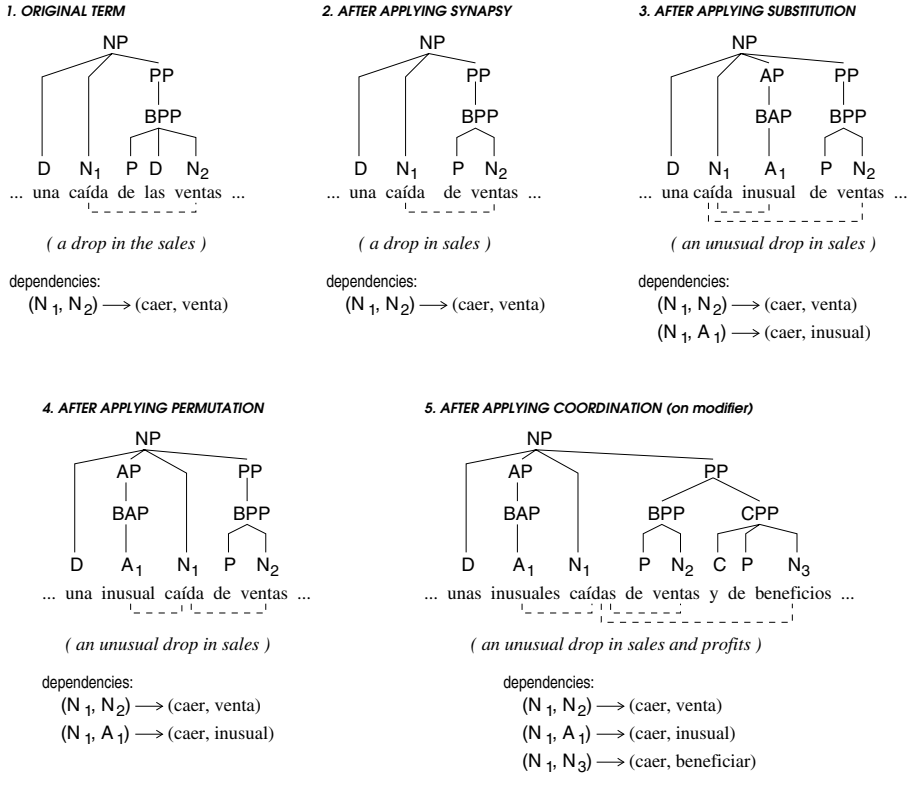


Fig. 1. Syntactic variants of *una caída de las ventas* (a drop in the sales)

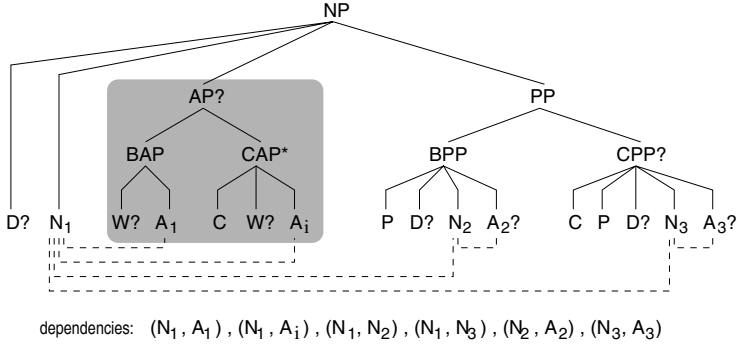


Fig. 2. Syntactic pattern for the syntactic variants of *una caída de las ventas*

and its twin, in which the *adjective phrase* postposed to the head name (shaded) is placed before such name. We have followed the finite-state shallow parsing approach used in successful information extraction systems [1,9,10], instead of the full parsing approach applied by some information retrieval systems [16]. In our

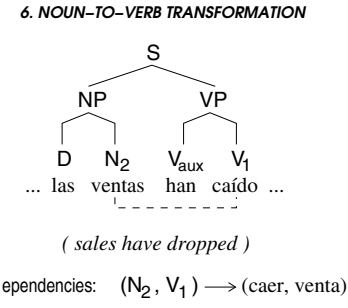


Fig. 3. Morpho-syntactic variant of *una caída de las ventas* (a drop in the sales)

particular example, the syntactic pattern is translated into the following regular expression, $D?\ N_1\ (W?\ A_1\ (C\ W?\ A_i))?\ P\ D?\ N_2\ A_2?\ (C\ P\ D?\ N_3\ A_3?)?$, keeping its associated syntactic dependency pairs. In this way, we can identify and extract multi-word index terms through simple pattern matching over the output of the tagger/lemmatizer, dealing with the problem from a surface processing approach at lexical level, leading to a considerable reduction of the running cost.

2.2 Morpho-syntactic Variants

Morpho-syntactic variants are classified according to the nature of the derivational transformations applied to their words:

- *Iso-categorial:* morphological derivation process does not change the category of words, but only transforms one noun syntagma into another. There are two possibilities: noun-to-noun and adjective-to-adjective.
- *Hetero-categorial:* morphological derivation does result in a change of the category of a word. There are also two possibilities: noun-to-verb and noun-to-adjective.

Figure 3 shows the structure of the sentence *las ventas han caído* (sales have dropped), a morpho-syntactic variant of *una caída de las ventas* (a drop in the sales) that involves a noun-to-verb transformation: *caída* (drop) becomes *caer* (to drop). We can observe that the words in the dependency pair extracted are the nouns *caída* (drop) and *ventas* (sales) in the source phrase, and the verb *caer* (to drop) and the noun *ventas* (sales) in its variant; therefore, different syntactic dependency pairs would be obtained. However, there exist a derivational relation between *caer* and *caída* and so, by employing morphological families to conflate the components of syntactic dependency pairs, we are able to obtain the same pair for both the source phrase and the variant. Therefore, common information to both the variant and the original term is conflated in the same way, which is the objective pursued [2].

Finally, we must remark that syntactic variants involve inflectional morphology but not derivational morphology, whereas morpho-syntactic variants involve

both inflectional and derivational morphology. In addition, syntactic variants have a very restricted scope (the noun phrase) whereas morpho-syntactic variants can span a whole sentence, including a verb and its complements, as in the case of Fig. 3.

3 Evaluation

The lack of a standard evaluation corpus has been a great handicap for the development of IR research in Spanish.[1] This situation is changing due to the incorporation in CLEF-2001 [17] of a Spanish corpus which is expected to become a standard. The techniques proposed in this paper have been integrated very recently, and therefore we could not participate in the 2001 edition, but we have joined CLEF competition in 2002, a fact that has allowed us to access the document collection and queries of the previous edition. Thus, the results reported in this section correspond to non-official experiments.[2]

The Spanish CLEF corpus is formed by 215,738 documents corresponding to the news provided by EFE, a Spanish news agency, in 1994. Documents are formatted in SGML, with a total size of 509 Megabytes. After deleting SGML tags, the size of the text corpus is reduced to 438 Megabytes. Each query consists of three fields: a brief title statement, a one-sentence description, and a more complex narrative specifying the relevance assessment criteria. We have employed the three fields to build the final query submitted to the system.

The conflation of multi-word terms by means of the extraction of syntactic dependency pairs from queries and documents is independent of the indexing engine and so, any standard text indexing engine may be employed. Nevertheless, each engine will behave according to its own characteristics, such as indexing model, ranking algorithm, etc. [19]. The results we show here have been obtained with SMART, using the `ltc-lnc` weighting scheme [4], without relevance feedback.

We have compared the results obtained by four different indexing methods:

- Stemmed text after eliminating stopwords (*stm*). In order to apply this technique, we have tested several stemmers for Spanish. Finally, the best results we obtained were for the stemmer used by the open source search engine Muscat[3], based on Porter's algorithm [3].

[1] The test collection used in the Spanish track of TREC-4 (1995) and TREC-5 (1996), formed by news articles written in Mexican-Spanish, is no longer freely available.

[2] We have also tested some of the techniques proposed in this article over our own, non standard, corpus, formed by 21,899 news articles (national, international, economy, culture,...) with an average length of 447 words, considering a set of 14 natural language queries with an average length of 7.85 words per query, 4.36 of which were content words. Results are reported in [19].

[3] Currently, Muscat is not an open source project, and the web site `http://open.muscat.com` used to download the stemmer is not operating. Information about a similar stemmer for Spanish (and other European languages) can be found at `http://snowball.sourceforge.net/spanish/stemmer.html`.

Table 1. Number of index terms extracted from the CLEF corpus

	plain text	***stm***	***lem***	***fam***	***f-sdp***
Total	68,530,085	33,712,903	33,158,582	33,158,582	58,497,396
Unique	529,914	345,435	388,039	384,003	5,129,665

Table 2. Performance measures

	stm	***lem***	***fam***	***f-sdp***
Documents retrieved	49,000	49,000	49,000	49,000
Relevant documents retrieved	2,576	2,554	2,563	2,565
R-precision	0.4787	0.4809	0.4814	0.4692
Average precision per query	0.4915	0.4749	0.4843	0.4669
Average precision per relevant docs	0.5561	0.5521	0.5492	0.5189
11-points average precision	0.4976	0.4864	0.4927	0.4799

- Conflation of content words (nouns, adjectives and verbs) via lemmatization (*lem*), i.e. each form of a content word is replaced by its lemma. This kind of conflation takes only into account inflectional morphology.
- Conflation of content words by means of morphological families (*fam*), i.e. each form of a content word is replaced by the representative of its morphological family. This kind of conflation takes into account both inflectional and derivational morphology.
- Text conflated by means of the combined use of morphological families and syntactic dependency pairs (*f-sdp*).

The methods *lem*, *fam*, and *f-sdp* are linguistically motivated. Therefore, they are able to deal with some complex linguistic phenomena such as clitic pronouns, contractions, idioms, and proper name recognition. In contrast, the method *stm* works simply by removing a given set of suffixes, without taking into account such linguistic phenomena, yielding incorrect conflations that introduce noise in the system. For example, clitic pronouns are simply considered a set of suffixes to be removed. Moreover, the employment of finite-state techniques in the implementation of our methods let us to reduce their computational cost, making possible their application in practical environments.

Table 1 shows the statistics of the terms that compose the corpus. The first and second row show the total number of terms and unique terms obtained for the indexed documents, respectively, either for the source text and for the different conflated texts. Table 2 shows performance measures as defined in the standard `trec_eval` program. The monolingual Spanish task in 2001 considered a set of 50 queries, but for one query any relevant document exists in the corpus, and so the performance measures are computed over 49 queries. Table 3 shows in its left part the precision attained at the 11 standard recall levels. We can observe that linguistically motivated indexing techniques beats *stm* for

Table 3. Average precision at 11 standard recall levels and at N seen documents

Recall	Precision				N	Precision			
	stm	***lem***	***fam***	***f-sdp***		***stm***	***lem***	***fam***	***f-sdp***
0.00	0.8426	0.8493	0.8518	0.8658	5	0.6122	0.6204	0.6367	0.5918
0.10	0.7539	0.7630	0.7491	0.7422	10	0.5551	0.5245	0.5429	0.5143
0.20	0.6971	0.6738	0.6895	0.6766	15	0.5075	0.4871	0.4925	0.4612
0.30	0.6461	0.6117	0.6312	0.6047	20	0.4735	0.4500	0.4510	0.4398
0.40	0.5669	0.5589	0.5656	0.5305	30	0.4238	0.4136	0.4095	0.3980
0.50	0.5013	0.4927	0.4979	0.4687	100	0.2827	0.2759	0.2769	0.2661
0.60	0.4426	0.4209	0.4252	0.4211	200	0.1893	0.1903	0.1877	0.1813
0.70	0.3832	0.3636	0.3641	0.3444	500	0.0979	0.0969	0.0970	0.0952
0.80	0.3221	0.3080	0.3109	0.2941	1000	0.0526	0.0521	0.0523	0.0523
0.90	0.2140	0.2109	0.2221	0.2113					
1.00	0.1037	0.0974	0.1126	0.1194					

low levels of recall. This fact means that more highly relevant documents are placed in the top part of the ranking list applying these techniques. As a complement, the right part of Table 3 shows the precision computed at N seen documents.

4 Conclusion

In this article we have studied how the extraction of head-modifier pairs impacts the performance of text retrieval systems. Albeit our scheme is oriented towards the indexing of Spanish texts, it is also a proposal of a general architecture that can be applied to other languages with slight modifications. We have tested our approach with the CLEF 2001 collection of documents and queries, and the results of our experiments are consistent with the results obtained for English and Germanic languages by other IR systems based on NLP techniques [15,13,14,16]. As in [15], syntax does not improve average precision, but is the best technique for low levels of recall. A similar conclusion can be extracted from the work of [13] on Dutch texts, where syntactic methods only beats statistical ones at low levels of recall. Our results with respect to syntactic dependency pairs seem to be better that those of Perez-Carballo and Strzalkowski [16]. It is difficult to know if this improvement is due to a more accurate extraction of pairs or due to differences between Spanish and English constructions.

An important characteristic of the CLEF collection that can have a considerable impact on the performance of linguistically motivated indexing techniques is the large number of typographical errors present in documents, as have been reported in [8]. In particular, titles of the news (documents) are in capital letters without accents. We must take into account that the title of a news article is usually very indicative of its topic.

References

1. C. Aone, L. Halverson, T. Hampton, and M. Ramos-Santacruz. SRA: Description of the IE^2 system used for MUC-7. In *Proc. of the MUC-7*, 1998.
2. A. Arampatzis, T. van der Weide, C. Koster, and P. van Bommel. Linguistically motivated information retrieval. In *Encyclopedia of Library and Information Science*. Marcel Dekker, Inc., New York and Basel, 2000.
3. R. Baeza-Yates and B. Ribeiro-Neto. *Modern information retrieval*. Addison-Wesley, Harlow, England, 1999.
4. C. Buckley, J. Allan, and G. Salton. Automatic routing and ad-hoc retrieval using SMART: TREC 2. In D. K. Harman, editor, *Proc. of TREC-2*, pages 45–56, Gaithersburg, MD, USA, 1993.
5. J. Carrol, T. Briscoe, and A. Sanfilippo. Parser evaluation: a survey and a new proposal. In *Proc. of LREC'98*, pages 447–454, Granada, Spain, 1998.
6. M. Dillon and A. S. Gray. FASIT: A fully automatic syntactically based indexing system. *Journal of the American Society for Information Science*, 34(2):99–108, 1983.
7. J. L. Fagan. Automatic phrase indexing for document retrieval: An examination of syntactic and non-syntactic methods. In *Proc. of SIGIR'87*, pages 91–101, 1987.
8. C. G. Figuerola, R. Gómez, A. F. Zazo, and J. L. Alonso. Stemming in Spanish: A first approach to its impact on information retrieval. In [17].
9. R. Grishman. The NYU system for MUC-6 or where's the syntax? In *Proc. of MUC-6*. Morgan Kaufmann Publishers, 1995.
10. J. R. Hobbs, D. Appelt, J. Bear, D. Israel, M. Kameyama, M. Stickel, and M. Tyson. FASTUS: A cascaded finite-state transducer for extracting information from natural-language text. In E. Roche and Y. Schabes, editors, *Finite-State Language Processing*. MIT Press, Cambridge, MA, USA, 1997.
11. C. Jacquemin and E. Tzoukermann. NLP for term variant extraction: synergy between morphology, lexicon and syntax. In T. Strzalkowski, editor, *Natural Language Information Retrieval*, pages 25–74. Kluwer Academic Publishers, Dordrecht/Boston/London, 1999.
12. J. S. Justeson and S. M. Katz. Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, 1:9–27, 1995.
13. W. Kraaij and R. Pohlmann. Comparing the effect of syntactic vs. statistical phrase indexing strategies for Dutch. In C. Nicolaou and C. Stephanidis, editors, *Research and Adavanced Technology for Digital Libraries*, volume 1513 of *LNCS*, pages 605–614. Springer-Verlag, Berlin/Heidelberg/New York, 1998.
14. B.-K. Kwak, J.-H. Kim, G. Lee, and J. Y. Seo. Corpus-based learning of compound noun indexing. In J. Klavans and J. Gonzalo, editors, *Proc. of the ACL'2000 workshop on Recent Advances in Natural Language Processing and Information Retrieval*, Hong Kong, October 2000.
15. M. Mittendorfer and W. Winiwarter. Exploiting syntactic analysis of queries for information retrieval. *Data & Knowledge Engineering*, 2002.
16. J. Perez-Carballo and T. Strzalkowski. Natural language information retrieval: progress report. *Information Processing and Management*, 36(1):155–178, 2000.
17. C. Peters, editor. *Working Notes for the CLEF 2001 Workshop*. Darmstadt, Germany, 2001. Available at `http://www.clef-campaign.org`.

18. J. Vilares, D. Cabrero, and M. A. Alonso. Applying productive derivational morphology to term indexing of Spanish texts. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 2004 of *LNCS*, pages 336–348. Springer-Verlag, Berlin-Heidelberg-New York, 2001.
19. J. Vilares, M. Vilares, and M. A. Alonso. Towards the development of heuristics for automatic query expansion. In H. C. Mayr, J. Lazansky, G. Quirchmayr, and P. Vogel, editors, *Database and Expert Systems Applications*, volume 2113 of *LNCS*, pages 887–896. Springer-Verlag, Berlin-Heidelberg-New York, 2001.

Using Latent Semantic Indexing as a Measure of Conceptual Association for Noun Compound Disambiguation

Alan M. Buckeridge and Richard F.E. Sutcliffe

Department of Computer Science and Information Systems
University of Limerick, Limerick, Ireland
{Alan.Buckeridge,Richard.Sutcliffe}@ul.ie

Abstract. Noun compounds are a frequently occurring yet highly ambiguous construction in natural language; their interpretation relies on extra-syntactic information. Several statistical methods for compound disambiguation have been reported in the literature; however, a striking feature of all these approaches is that disambiguation relies on statistics derived from unambiguous compounds in training, meaning they are prone to the problem of sparse data. Other researchers have overcome this difficulty somewhat by using manually crafted knowledge resources to collect statistics on "concepts" rather than noun tokens, but have sacrificed domain-independence by doing so. We report here on work investigating the application of Latent Semantic Indexing [4], an Information Retrieval technique, to the task of noun compound disambiguation. We achieved an accuracy of 84%, indicating the potential of applying vector-based distributional information measures to syntactic disambiguation.

1 Introduction

Noun compounds are a frequently encountered construction in natural language processing (NLP), consisting of a sequence of two or more nouns which together function syntactically as a noun. In English, compounds consisting of two nouns are mostly right-headed; however, compound construction is recursive and both the modifier and the head can themselves be compounds, resulting in structural ambiguities. Consider the following pair of noun compounds:

1. a) (*styrofoam* (*coffee cup*))
   b) ((*computer science*) *student*)

Both compounds consist of the same parts-of-speech, yet the structures differ: (1a) is right-branching, while (1b) is left-branching.

Phrase structure grammar rules for noun compounds are often similar in form to $NP \rightarrow NP\ NP$ [7]. This rule is applied once to two-word noun compounds, and recursively in the case of longer compounds; therefore the syntax of compounds longer than two words is underconstrained by grammar, resulting in a syntactic ambiguity which grows exponentially with the length of the compound.

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 12–19, 2002.

Besides causing problems for syntactic parsers, the ambiguities inherent in these structures pose difficulties for NLP systems which attempt to analyse the underlying semantic relationships present in noun compounds. Often, the first step in such analyses is to decompose noun compounds into nested modifier-head pairs [1]. However, such a decomposition is non-trivial for the case of compounds consisting of three or more nouns due to the structural ambiguity of these constructions.

The identification of modifier-head pairs in compounds also has applications within the field of information retrieval (IR). Extracting modifier-head pairs from text and including these as compound indexing terms can improve recall and precision [5,16]. However, obtaining good modifier-head pairs is once again hampered by "the notorious ambiguity of nominal compounds" [16, p.107].

To summarise, the syntactic disambiguation of noun compounds is important for several NLP applications; however, disambiguation is difficult because attachments within compounds are not syntactically governed. Clearly, then, this lack of syntactic constraints forces us to consider the use of extra-syntactic factors in the process of disambiguation. The work reported here describes a system for automatically deriving a syntactic analysis of noun compounds by adapting a well-established IR technique to supply this extra-syntactic information.

The remainder of this paper is organised as follows: Section 2 describes previous work using corpus statistics and lexical-semantic information to disambiguate noun compounds. Section 3 briefly discusses Latent Semantic Indexing, the IR technique which we apply to the problem of noun compound disambiguation. Section 4 reports on an experiment using this technique. Section 5 concludes the paper and discusses future work.

2 Previous Work

The majority of corpus statistical approaches to compound disambiguation use a variation of what Lauer [7] refers to as the *adjacency algorithm*. This algorithm was originally proposed by Marcus [10], and essentially operates by comparing the acceptability of immediately adjacent noun pairs. Specifically, given a sequence of three nouns *n1 n2 n3*, if (*n2 n3*) is a more acceptable constituent than (*n1 n2*), then build (*n1* (*n2 n3*)); else build ((*n1 n2*) *n3*).

There remains the question of how "acceptability" is to be determined computationally. Most studies (e.g., [1,5,14,16]) collect statistics on the occurrence frequency of structurally unambiguous subcomponents to inform the analysis of the ambiguous compound. For example, given the compound "font dialog box", the structure (*font* (*dialog box*)) would be preferred if (*dialog box*) occurred more frequently than (*font dialog*) in the corpus. However, by assuming that sufficient examples of subcomponents exist in the training corpus, all the above approaches risk falling foul of the *sparse data problem*. Most noun-noun compounds are rare, and statistics based on such infrequent events may lead to an unreliable estimation of the acceptability of particular modifier-head pairs.

The work of Resnik [15] goes some way towards alleviating this problem. Rather than collecting statistics on individual words, he instead counts co-occurrences between *concepts* (represented by WordNet synsets [13]) and nouns. These statistics are used to calculate *selectional association*, a measure motivated by information theory (see [15] for full details). "Acceptability" in the adjacency algorithm is then measured in terms of the selectional association between a modifier and head. Of a sample of 156 three-noun compounds drawn from the *Wall Street Journal* corpus in the Penn Treebank, his method achieved 72.6% disambiguation accuracy.

Lauer [7] similarly generalises from individual nouns to semantic classes or concepts, deriving his classes from categories in the 1911 edition of Roget's Thesaurus. Similar to Resnik, Lauer extracts a training set of unambiguous noun-noun modifier-head compounds to estimate the degree of association between Roget categories. He calls this measure *conceptual association*, and uses this to calculate the acceptability of noun pairs for the disambiguation of three-noun compounds. However, his approach differs from most others in that he does not use the adjacency algorithm, instead using a method motivated by dependency grammar. Lauer's *dependency algorithm* operates as follows: Given a three-noun compound *n1 n2 n3*, if (*n1 n3*) is more acceptable than (*n1 n2*), then build (*n1* (*n2 n3*)); else build ((*n1 n2*) *n3*). Lauer discusses how the adjacency and dependency algorithms differ in terms of the proportion of left and right branching analyses they predict. According to the dependency model, the ratio of left to right branching will be 2:1; in contrast, the adjacency model predicts equal proportions of left and right branching compounds. Thus the dependency algorithm's predictions are more in line with experimental findings, which show that approximately two-thirds of compounds are left branching [7] (though see [1] for contradictory evidence).

Lauer tested both the dependency and adjacency algorithms on a set of 244 three-noun compounds extracted from an online edition of Grolier's Encyclopedia and found that the dependency algorithm consistently outperformed the adjacency algorithm, achieving a maximum of 81% accuracy on the task. Overall, he found that estimating the parameters of his probabilistic model based on the distribution of concepts rather than that of individual nouns resulted in superior performance, thus providing further evidence of the effectiveness of conceptual association in noun compound disambiguation.

All of the above approaches rely on a variation of finding subconstituents elsewhere in the corpus and using these to decide how the larger, more ambiguous compounds are structured. However, there is always the possibility that these systems might encounter modifier-head pairs in testing which never occurred in training, forcing the system to "back off" to some default strategy. Nor is this problem is eliminated in the work of Resnik [15] and Lauer [7] where statistics are collected on pairs of *concepts* rather than pairs of noun tokens. Moreover, the methods of Resnik and Lauer both depend on hand-crafted knowledge sources; the applicability of their approaches is therefore limited by the coverage of these resources. Therefore, it would be preferable to have a method of measuring

conceptual associations which is less domain-dependent; this prompted us to investigate whether Latent Semantic Indexing might satisfy these requirements.

3 Brief Description of Latent Semantic Indexing

Latent Semantic Indexing (LSI) is a variant of the vector-space approach in information retrieval. It takes as input a collection of documents, from which it constructs an $m \times n$ word-document matrix A; cell a_{ij} of the matrix denotes the frequency with which term i occurs in document j. The core of LSI is its use of *singular value decomposition* (SVD), a mathematical technique closely related to eigenvector decomposition and factor analysis. SVD factors the matrix A into the product of three matrices: $A = U\Sigma V^T$. U and V contain the left and right singular vectors of A, respectively, while Σ is a diagonal matrix containing the singular values of A in descending order. By retaining only the k largest singular values and setting the remaining smaller ones to zero, we obtain a new diagonal matrix Σ_k; then the product of $U\Sigma_k V^T$ is the $m \times n$ matrix A_k which is only approximately equal to A. This truncated SVD re-represents the word-document relationships in A using only the axes of greatest variation, in effect compressing and smoothing the data in A. It is claimed that this compression step captures important regularities in the patterns of word co-occurrences while ignoring smaller variations that may be due to idiosyncrasies in the word usage of individual documents. The result of condensing the matrix in this way is that words which occur in similar documents will be represented by similar vectors, even if these words never actually co-occur in the same document. Thus it is claimed that LSI captures deeper associative relationships than mere word-word correlations.

Retrieval proceeds similarly to the traditional vector-space model, but using the $m \times n$ matrix A_k rather than the original word-document matrix A. Results have shown LSI's performance can often be superior to that of the standard vector-space model [4]. Critical for success is the choice of value for k. Unfortunately, the value of k which gives the optimal retrieval performance can only be determined empirically, and will depend on the particular application (see [2, 4] for more thorough discussions of SVD and its application to information retrieval).

Because word vectors are originally based on their distribution of occurrence across documents, each vector can be interpreted as a summary of a word's contextual usage; words are thus similar to the extent that they occur in similar contexts. There is a growing body of literature indicating that distributional information of the kind captured by LSI plays an important role in various aspects of human cognition. For example, people's similarity judgements of marginally familiar and nonsense words have been shown to be affected by the distributional properties of the contexts in which they were read, supporting the role of distributional information in developing representations of word meaning [12]. Other language processing behaviours in which distributional information has been implicated include lexical priming [3,9] and synonym selection [6].

For the work reported here, the most interesting aspect of distributional information is its purported ability to model conceptual categorisation. Several studies (e.g., [3,8]) have shown that similarity between concepts can be measured quite successfully using simple vectors of contextual usage; results show that the performance of such systems correlates well with that of humans on the same tasks. These results are all the more impressive when we consider that such systems use no hand-coded semantic knowledge; the conceptual representations are derived automatically from training corpora.

Noun compound disambiguation appears to be an NLP application for which such measures of conceptual association would be useful. Both the adjacency and dependency algorithms described above in Sect. 2 rely on some measure of the "acceptability" of pairs of nouns to disambiguate noun compounds. Techniques such as LSI offer a simple, robust, and domain-independent way in which concepts and the associations between them can be represented. In the next section, we describe an experiment which looks at the efficacy of LSI's conceptual representations in disambiguating noun compounds.

4 Disambiguating Noun Compounds with LSI

4.1 Method

Preparation. The corpus used for our study was the *Lotus Ami Pro Word Processor for Windows User's Guide Release 3*, consisting of approximately 136,000 words spread over 704 sections. We first ran the LSI software on the corpus to create the word-by-document matrix. The software also subsequently performed singular value decomposition on the resulting 2132-words by 704-documents matrix. No stop-words were used, as we found that their inclusion degraded performance slightly.

To prepare the test compounds, we extracted sequences of three nouns from the corpus using a part-of-speech tagger [11] and Perl regular expressions. Sequences which were not true three-noun compounds were discarded. The remaining 307 noun compounds were bracketed manually and constituted our test set; some examples are shown in Table 1.

Table 1. Some example noun compounds taken from our test set. Each row shows an example of a manually bracketed compound, along with its branching

Noun Compound	Branching
((Ami Pro) document)	Left
(custom (line spacing))	Right
(data (dialog box))	Right
((Microsoft Windows) applications)	Left
(windows (program manager))	Right
((zip code) information)	Left

Procedure. Both the adjacency and dependency models were investigated (see Sect. 2). Recall that the adjacency algorithm operates by comparing the acceptability of (*n1 n2*) and (*n2 n3*), whereas the dependency algorithm compares the acceptability of (*n1 n2*) and (*n1 n3*). "Acceptability" in our approach was measured by calculating the cosine of the angle between each pair of word vectors; a higher cosine indicated a stronger association between each word in a pair. In the case of a tie, a left branching analysis was preferred, as the literature suggests that this is the more common structure [7,15]. Thus a default strategy of always guessing a left branching analysis served as the baseline in this study.

As we could not tell beforehand what the optimal value of k would be (see Sect. 3 above), we used a range of factor values. The values used were 10–700 in steps of 10, and the full 704 factors. For each factor value, we obtained the percentage accuracy of both the adjacency and dependency models.

4.2 Results

The results of the experiment are summarised in Fig. 1. The figure clearly shows that the adjacency algorithm substantially outperformed the dependency algorithm. The greatest overall accuracy was 84%, obtained using the adjacency algorithm and 240 SVD factors. The greatest accuracy achieved for the dependency model was 71% using 230 SVD factors. As the task involved choosing the best binary bracketing for a noun compound, we would expect an accuracy of 50% by chance. Of the 307 test compounds, 180 (59%) were left branching; thus, both algorithms performed substantially better than the baseline. These results compare favourably with those of Resnik [15] and Lauer [7] (73% and 81%, re-

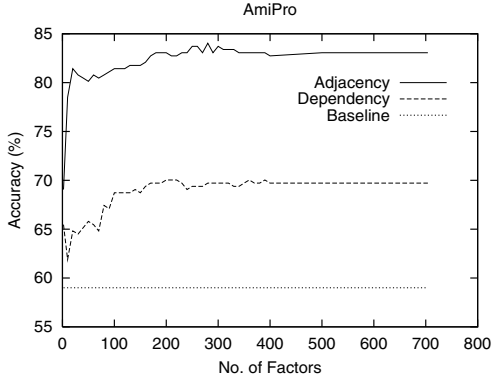


Fig. 1. Results of an experiment investigating noun compound disambiguation using LSI. The figure shows percentage disambiguation accuracy of the *adjacency* and *dependency* algorithms for a range of SVD factors, and clearly illustrates the superior performance of the adjacency algorithm

spectively); however, as their studies were conducted on different corpora, it would be imprudent to make direct comparisons at this stage.

Table 2 displays the distribution of left and right branching compounds in the test set. The table shows that the dependency algorithm assigned a considerably higher proportion of left branching analyses to the test set than the adjacency algorithm, a result which supports Lauer's [7] predictions about the dependency algorithm's behaviour (see Sect. 2 above). Clearly, this bias in favour of left branching analyses contributed to the inferior performance of the dependency algorithm in our study. Given a test set in which the proportion of actual left branching compounds was higher, it would be interesting to investigate whether we would still observe such a marked disparity in the algorithms' performance.

5 Conclusions and Future Research

The results reported here are encouraging; the highest accuracy of 84% indicates the potential of our approach. We therefore intend to pursue our investigation of the utility of applying vector-based measures of semantic similarity to the problem of syntactic disambiguation. An attractive feature of this approach is that it requires no manually constructed knowledge sources, meaning that it does not suffer the same coverage limitations as the methods of Lauer [7] and Resnik [15]; in principle, our approach can be applied to any domain.

We are currently examining the use of other techniques for deriving vector-based measures of conceptual association; preliminary investigations using a "sliding window" method [3,8] to disambiguate compounds from the *Ami Pro* corpus show results even better than those reported here. Present work involves setting various parameters to study their effect on performance. We are continuing to test both the adjacency and dependency algorithms, and have consistently found better performance using the former.

Future work will involve testing the technique in other domains; we also intend training on larger and more diverse corpora. Furthermore, we plan to investigate other examples of syntactic ambiguity, such as prepositional phrase attachment. Such structures pose many problems for traditional NLP systems, but may prove amenable to the techniques discussed in this paper.

Table 2. The distribution of left and right branching analyses in our experiment. The first row shows the correct proportion of left and right branching structures in the set of test compounds ($n = 307$). The second and third rows show the proportion of left and right analyses assigned by the adjacency and dependency algorithms

	Left	Right
Actual	180 (59%)	127 (41%)
Adjacency	171 (56%)	136 (44%)
Dependency	218 (71%)	89 (29%)

Acknowledgements. We thank Pat Hickey of the Department of Electronic and Computer Engineering, University of Limerick, for technical assistance, and two anonymous reviewers for helpful comments.

References

[1] Barker, K.: A trainable bracketer for noun modifiers. In *Proceedings of the Twelfth Canadian Conference on Artificial Intelligence*, pages 196–210, Vancouver, 1998.

[2] Berry, M. W., Dumais, S. T., Letsche, T. A.: Computational methods for intelligent information access. In *Proceedings of Supercomputing '95*, San Diego, CA, 1995.

[3] Burgess, C., Lund, K.: The dynamics of meaning in memory. In E. Dietrich and A. Markman, editors, *Cognitive Dynamics: Conceptual and Representational Change in Humans and Machines*, pages 17–56. Lawrence Erlbaum Associates Inc., Hillsdale, NJ, 1999.

[4] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., Harshman, R.: Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.

[5] Evans, D. A., Zhai, C.: Noun-phrase analysis in unrestricted text for information retrieval. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 17–24, Santa-Cruz, CA, June 1996.

[6] Landauer, T. K., Dumais, S. T.: A solution to Plato's problem: The Latent Semantic Analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review*, 104:211–240, 1997.

[7] Lauer, M.: *Designing Statistical Language Learners: Experiments on Noun Compounds*. PhD thesis, Macquarie University, Sydney, Australia, 1995.

[8] Levy, J. P., Bullinaria, J. A.: Learning lexical properties from word usage patterns: Which context words should be used? In *Proceedings of the Sixth Neural Computation and Psychology Workshop*, pages 273–282. London: Springer, 2001.

[9] Lowe, W., McDonald, S.: The direct route: Mediated priming in semantic space. In *Proceedings of the 22nd Annual Meeting of the Cognitive Science Society*, pages 806–811. Lawrence Erlbaum Associates, 2000.

[10] Marcus, M.: *A Theory of Syntactic Recognition for Natural Language*. MIT Press, Cambridge, MA, 1980.

[11] Mason, O.: `http://www.english.bham.ac.uk/staff/oliver/software/tagger/`.

[12] McDonald, S., Ramscar, M.: Testing the distributional hypothesis: The influence of context on judgements of semantic similarity. In *Proceedings of the 23rd Annual Conference of the Cognitive Science Society*, 2001.

[13] Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., Miller, K.: Five papers on WordNet. Technical Report 43, Cognitive Science Laboratory, Princeton University, July 1990.

[14] Pustejovsky, J., Bergler, S., Anick, P.: Lexical semantic techniques for corpus analysis. *Computational Linguistics*, 19(2):331–358, 1993.

[15] Resnik, P. S.: *Selection and Information: A Class-Based Approach to Lexical Relationships*. PhD thesis, University of Pennsylvania, 1993.

[16] Strzalkowski, T., Vauthey, B.: Information retrieval using robust natural language processing. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 104–111, Newark, Delaware, USA, 1992.

RADAR: Finding Analogies Using Attributes of Structure

Brian P. Crean[1] and Diarmuid O'Donoghue[2]

[1]Galway-Mayo Institute of Technology, Castlebar Campus
Westport Road, Castlebar, Co.Mayo, Ireland also [2]
Brian.crean@gmit.ie
[2]Department of Computer Science, NUI Maynooth,
Co. Kildare, Ireland.
Diarmuid.odonoghue@may.ie
http://www.cs.may.ie/~dod/

Abstract. RADAR is a model of analogy retrieval that employs the principle of systematicity as its primary retrieval cue. RADAR was created to address the current bias toward semantics in analogical retrieval models, to the detriment of structural factors. RADAR *recalls* 100% of structurally identical domains. We describe a technique based on "derived attributes" that captures structural descriptions of the domain's *representation* rather than domain contents. We detail their use, recall and performance within RADAR through empirical evidence. We contrast RADAR with existing models of analogy retrieval. We also demonstrate that RADAR can retrieve both semantically related and semantically unrelated domains, even without a complete target description, which plagues current models.

1 Introduction

An analogy is a structure-based comparison between a problem domain (the target) and some known domain (the source). Existing similarities form the basis for transferring additional information from the source to the target domain. *"So, IF the nucleus is like the sun and the planets orbit the sun, THEN the electrons must also revolve about the nucleus"*. Yet much research has focused on interpreting predetermined analogies rather than discovering new candidate analogies.

Analogy retrieval is an essential yet complex process, where numerous similarity constraints (semantic, structural and pragmatic) can be considered in judging the usefulness of a source/target pairing [1]. The retrieval process is further complicated when one considers that cross-domain analogies need not share superficial semantic features between the participating domains. Structural similarity has received a great deal of acceptance in analogical mapping, yet has had startlingly little impact on analogy retrieval models. We believe that it is not unreasonable to include structural considerations within the retrieval process itself. Systematicity [2] would allow a deliberate broadening of knowledge to reach retrieval and subsequent stages, and thereby form new analogies

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 20-27, 2002.

Frequently, domains lacking obvious semantic resemblance can be used to form useful analogies [3], for example *"Nuclear war is like a game of tic-tac-toe*" (adopted from the motion picture "War Games"). Taking a recognised domain like tic-tac-toe that frequently ends in a draw (no winner) can lead to recognising the futility of nuclear war i.e. no winner. In order to interpret a possible candidate analogy we first must discover it. Yet in current models, retrieval is governed by the semantic content (meaning) of entities common to both analogs. Such systems can only retrieve domains that are represented similarly - and the retrieval of structurally similar domains is purely accidental. It is merely a happy coincidence that any semantically similar domains ever have the same structure as each other. A crucial consequence of these inflexible retrieval models, is their inability to explain the role of analogy in creativity, inspiration and insight as identified by Boden [4].

Initially, we want to retrieve a *structurally* identical source domains, given some target domain problems. However, analogies are typically formed between a smaller target domain and a larger source - since the source must supply additional information to the target domain. Therefore, not only are we looking for *isomorphic* source domains, but we also wish to identify *homomorphic* sources.

These factors will undoubtedly increase the possibility of retrieving inappropriate source domains. Current retrieval models reject many inappropriate sources by their inability to form a large mapping. However, our technique could use the analogical validation phase to reject inappropriate comparisons. A model of structural retrieval would also have to operate in a computationally tractable manner. In this paper we introduce the RADAR (Retrieving Analogies utilising Derived AttRibutes) model and its method of encoding structural traits of domains through derived attributes. We detail its operation, and how it improves over current analogy retrieval models.

2 Existing Analogy Retrieval Models

Before we discuss models of analogy retrieval we briefly describe our objectives from an information retrieval perspective. As we shall see, existing models have very poor *recall,* because they either ignore or severely dis-favour semantically distant domains. Thus, they are impotent from the perspective of generating creative analogies. Furthermore, the *precision* of some models is very poor because semantic similarity is independent of the structural similarity required to form an analogy. We wish to explain how structurally similar domains, even ones that are semantically un-related to the target, might be retrieved. Such comparisons underlie many breakthroughs in science - *"the heart is a pump, the brain is a telephone networks, light is a wave vs. light is a particle"*. Crucially, no current model of analogical retrieval can explain how a target domain can cause the retrieval of a semantically unrelated source. Consider the following target sentence (taken from [5]), where Felix is a cat and Mort is a mouse.

T_1 : "*Felix bit Mort causing Mort to flee from Felix"*

We wish to find domains that are *structurally* similar to T_1, so that we can later identify any cross-domain analogies. So, for the given source we might wish to identify a candidate source domain such as S_2, where Mary is a woman and Frank is a man**.**

S_2 : *"Mary seduced Frank causing Frank to kiss Mary"*

Of course, most cross-domain analogies will be invalid but we must at least have the potential to retrieve them. Identifying the useful analogies from these cross-domain comparisons is the task for the later *mapping* and *validation* phases.

2.1 Analogy Retrieval Models

With domains described by content vectors, MAC/FAC [6] creates a skeletal description by counting the number of times distinct predicates and objects occur in a representation. Estimating the structural overlap between source and target is presented by a dot product computation on the respective content vectors. The highest dot product and all those within a 10 % range are deemed structurally similar. Crucially, MAC/FAC cannot retrieve semantically distant domains due to weak predicate and object similarity in the content vectors. Its frailty is also evident when presented with a partial source and target description. Consider S_3 (additional information to S_2)

S_3 : *"Frank and Susanne are married. Mary seduced Frank causing Frank to kiss Mary, so Susanne divorced Frank".*

The respective T_1/S_3 content vector is weakened due to again the non-identicality of the domains on which MAC/FAC is dependant, signifying it is vulnerable when dealing with partial descriptions.

ARCS [7] uses tokenised semantic similarity between representations as preselection criteria in the creation of a parallel constraint satisfaction network. Structural correspondence is determined on the basis of isomorphic similarity between source(s) and target predicate arguments. ARCS uses a standard parallel connectionist relaxation algorithm to allow the network to settle. After the network settles sources are deemed suitable based on their activation strength. ARCS perseverance with semantic similarity as a filtering process means it casts semantically disparate domains aside from the outset, i.e. Felix the cat is not similar to Frank the man. In relation to the original T_1 and S_2, ARCS immediately rejects any possible retrieval due to weak semantic correspondence, the only recognised similarity is the higher-order predicate, *cause*. This indicates that ARCS is unable to retrieve semantically distant domains.

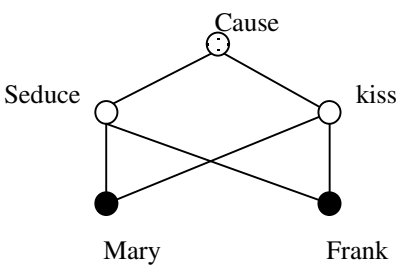
HRR's [5] combine semantic and structural similarity with an inclusive vector representation. Semantic identity is encoded into each entity in a representation. The structural traits of the representation are brought to the surface as role filler assignments (e.g. agent, patient) and are encoded along with each entity or predicate. HRR's structural retrieval is diminished by its tendency to favour more semantically similar source(s). Consider T_1 and S_2, the absence of semantic similarity between objects and predicates in both source and target, severely limits the possibility of semantically

distant domains being retrieved (also borne out by Plates original results [5]). The resultant "convolution" product (used by HRR's to estimate similarity) is considerably weak. HRR's suffer from a similar affliction as MAC/FAC in its inability to tolerate the loss of significant information. If presented with T_1 and S_3, the absence of identical structures between representations will weaken the measures of similarity even further. Though HRR's attempt to combine semantics and structure in retrieval, problems with non-identical structure and semantics are very problematic to HRR's.

Though CBR (Case-Based Reasoning) is similar to analogy retrieval in many respects, we exclude CBR from discussion in this paper as CBR is primarily concerned with intra-domain retrieval and is not concerned with retrieving semantically distinct domains.

3 Derived Attributes

One method to effect structural-based retrieval is to re-describe the domain description via micro-features. We use *derived attributes* to determine the structural characteristics of a domain representation. Derived attributes describe features of the representation itself, rather than qualities related to the real world. These structural attributes are not tailored to suit individual domains but are used to describe any conceivable domain. Each derived attribute describes a particular construct of the domain through a calculated value. A number of simple derived attribute types (Fig. 1) can be used to capture the structural characteristics of the domain's representation. Derived attributes such as the number of predicates and objects in a domain can convey simple structural detail. The identification of cyclic paths (loops) in the representation is also beneficial i.e. in Fig. 1, the path containing the entities *cause, seduce, frank* and *kiss* is an instance of a loop as it cycles back to the starting point. These derived attributes can distinguish between identically structured domains and similarity-structured domains - an important distinction as already highlighted in the operation of current models. For illustrative purposes the predicate representation of, S_2, *cause (seduce, kiss), seduce(mary, frank), kiss (frank, mary)* can be described by the derived attributes in Fig. 1.



Derived Attributes

Number of Loops : 3
Loop size : 4
Connectivity : 3
Number of predicates : 3
Number of atoms : 5

Fig. 1. Derived attribute description of S_2

The derived attributes depicted in Fig. 1 are just a subset of the structural composition of domains, other derived attribute types such as *number of roots*, *longest path* etc. can also be incorporated. These structural attributes are independent of any se-

mantic primitive descriptions housed in these domains and hence cannot be influenced by semantic considerations.

4 RADAR

We now describe RADAR (Retrieving Analogies using Derived AttRibutes), which bases analogy retrieval in the 'search space' of structural attributes. The premise of RADAR's retrieval algorithm is that domains with the same derived attributes must also have identical structure. Each domain in long-term memory is examined for its structural description, which generates various derived attributes types and their values. These values are stored in derived attributes stores and linked to the source in long-term memory.

When presented with a target analog, the target is also rendered into its own structural attributes, and the target's derived attributes activate the corresponding stores. All concepts in memory that have the same attribute value for a particular type will be identified through spreading activation. This indicates they are structurally identical for each structural feature. Activation values discriminate the strongest source(s) from dissimilarly structured analogs through a threshold level that rejects any concept below a certain similarity measure. The stronger the activation value, the more structurally identical a candidate source is to a target.

4.1 Identical Structure Retrieval

Presenting S_2 (source) and T_1 (target) from above, we demonstrate how RADAR retrieves identical structures. Retrieval is driven by the derived attributes of T_1, so any source that shares a corresponding derived attribute value will receive activation. RADAR successfully retrieves S_2 with and overall similarity metric of 100%. This 100 % signifies that both source and target share all derived attribute values and hence are structurally identical. This was as expected because the source representation is isomorphic with the target. Current models rely heavily on semantic similarity to guide retrieval but this demonstrates retrieval where semantic similarity is negilable, consequently are constrained by the targets semantic influence and unable to retrieve this valid analogy.

It also demonstrates RADAR's ability to retrieve creative analogies such as the *"Nuclear war is like a game of tic-tac-toe",* which again current models cannot retrieve. Identical results were obtained when presented other structurally identical domains that *do* share some semantic similarity, again Plate's "Spot bit Jane causing Jane to flee from Spot", Gick and Holyoak's [8] surgeon/general analogy, etc. These domains contain identical structural and RADAR, as expected, retrieves the identical sources based on identical structural attribute values.

4.2 Partial Structural Retrieval

This demonstration examines RADAR's ability to retrieve useful sources from memory, where the structural overlap between source and target is incomplete. (missing predicates and objects). This is vital as analogies are frequently used for learning, requiring that the source have more structure than the target - and consequently a different structure to it. Each source is broken down into its derived attribute pairings (Fig. 2) and stored in structure memory. RADAR again creates a basis for retrieval by analyzing the target, T_1 for its derived attribute values. Again, RADAR successfully retrieved the appropriate source, S_3, though with a smaller similarity scoring. As we can see from Fig. 2 the structure of the analogues do not match exactly, and this is reflected in the derived attribute values (again Fig. 2). But a subsection of the representations do share identical structure - the loops structure. Both share three loops of size four. The loop structures identify a structural similarity and brings it to the surface.

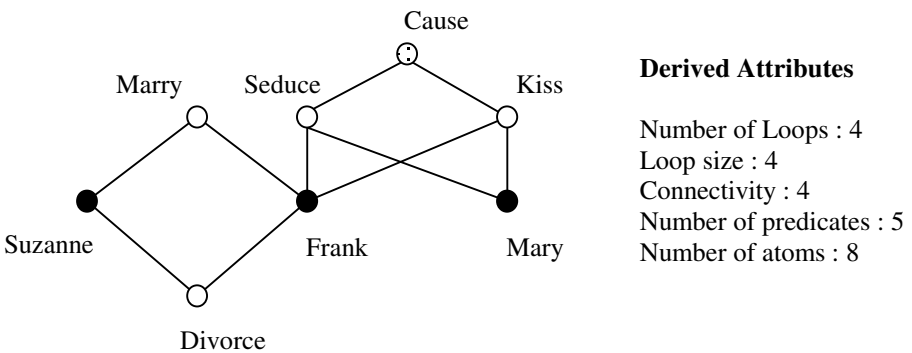


Fig. 2. Structural representation of the S_3's Love Triangle Story

We readily accept the argument that if more information were missing then retrieval accuracy would decay. This is of course a fact of retrieval, poor representations lead to miss-guided retrieval (if any). Remove the predicate *Divorce (Susanne, Frank)* from the representation, and the resultant derived attribute values would reflect this change. But we would argue that there does come a point in reminding, in the human cognitive process and cognitive modeling, when significant information is missing, retrieval will tend to be poor or not take place at all. This is perfectly analogous to the existing situation where semantically based retrieval performs poorly with missing information - but cannot retrieve outside its own domain. Likewise if more predicates or objects were added to the target, i.e. the proposition *move-in-with (Frank, Mary)* then the structure changes, but again there is a common substructure (loop structure). This experiment confirmed that RADAR operates successfully when presented with partial domains, on the provision that there exists some coherent structure between the representations. RADAR is the only model that considers partial source/target parings in retrieval.

5 Performance and Future Work

RADAR's overall performance was examined with a long-term memory containing frequently cited domains, chosen randomly from the analogy literature. In all seventy domains were stored. Domains were of varying complexity with an average of 8 predicates (ranging from a minimum of 1 to a maximum of 25 predicates) and 14 entities (also ranging from a minimum of 3 to a minimum of 39 entities) per domain. In the investigation, the largest loop construct considered was six. Selection is based on the highest scoring source domain, or group of sources joint highest retrieval score. Retrieval was classified as successful if the target caused the corresponding source identified in the literature to be retrieved.

RADAR retrieved the common matches on each occasion and significantly out performed other models in its ability to retrieve appropriate candidate source domains. RADAR retrieved an average of 4 sources (6%) when presented with a target. In each case the correct source was amongst the joint highest active sources. Significantly, RADAR can retrieve similar structured source when presented with identical and partial representations even when they share no semantic overlap between objects or predicates, where other models are deficient.

	MAC/FAC	ARCS	HRR	RADAR
Identical Structural Retrieval	Yes	Yes*	Yes +	Yes
Partial Structural Retrieval	No	Yes *	No	Yes
Identical Semantic distant Domain retrieval	No	No	No	Yes
Partial Semantic distant Domain retrieval	No	No	No	Yes

Table 1. Comparison of RADAR against common retrieval models

* on the provision that pre-selection will have semantic information
\+ retrieval accuracy will vary considerably from target

Derived attributes can be manipulated in order to re-describe the structural description of a representation and increase performance of the retrieval process. Similar to *weighted features* [9], where feature descriptors are weight based on their importance or usefulness, certain derived attributes can be marked as more relevant than others. Alternatively a nearest neighbour algorithm can be used to locate similar sources [O'Donoghue, Crean, in press]. Another technique is to simply increase the number of derived attributes used to describe a domain.

Domain retrieval using derived attributes is only as efficient as the derived attributes that supplement the raw domain information. Taking just five attribute types each with just 10 values, and making the best case assumption that our data is distributed

evenly along each value, then each location in derived attribute space would represent just 10 domains, for a base of 1,000,000 domains. This indicates the potential retrieval power of derived attributes. Of course, efficiency is increased with additional attribute types and values describing new structural qualities with particular utility to analogy retrieval.

6 Conclusion

We lay no claim that this is how structural retrieval is performed in human cognition. The focus of this work was on creating a computational model that is capable of structural domain retrieval in a computationally tractable manner - which overcomes the semantic restriction suffered by other models. We have demonstrated the ability of derived attributes in describing the structural make-up of domains. We then demonstrated their ability to retrieve semantically related and un-related domains, whether presented with a partial or complete target domain. We detailed these findings through the recall and precision performance of RADAR, using commonly cited domains from the analogy literature. RADAR successfully overcomes the limitations suffered by current retrieval models.

References

1. Eskeridge, T.C. "A Hybrid model of continuous analogical reasoning", In Branden (ed.), Advances in Connectionist and Neural Computation Theory, Norwood, NJ: Ablex, 1994
2. Gentner D "Structure-mapping: A theoretical framework for analogy", Cognitive Science, volume 7, pp. 155–170, 1983.
3. Kolodner, J. L. "Educational Implications of Analogy a view from Case-based Reasoning", American Psychologist, pp. 57–66 Volume 52, 1997.
4. Boden, M. A. "The Creative Mind", Abacus, 1994.
5. Plate Tony A., "Distributed Representations and Nested Compositional Structure", Graduate Department of Computer Science University of Toronto, 1994 Ph.D. Thesis.
6. Gentner D, Forbus D., " MAC/FAC : A model of Similarity-based Retrieval", pp 504–509, Proceedings of the 13th Conference Cognitive Science Society, 1991.
7. Thagard P., Holyoak K. J., Nelson G, Gochfield D., "Analog Retrieval by Constraint Satisfaction", pp. 259–310, Artificial Intelligence Volume 46, 1990.
8. Gick M. L. and Holyoak K. J., "Analogical Problem Solving", Cognitive Psychology, volume 12, pp. 306–355, 1980
9. Blum A. L., Langley. P. "Selection of relevant features and examples in machine learning". Artificial Intelligence volume 97, pp. 245–271, 1997.

Classifying Languages Based on Speech Rhythm

Fred Cummins

Department of Computer Science, University College Dublin
fred.cummins@ucd.ie
http://cspeech.ucd.ie/~{}fred

Abstract. Of all prosodic variables used to classify languages, rhythm has proved most problematic. Recent attempts to classify languages based on the relative proportion of vowels or obstruents have had some success, but these seem only indirectly related to perceived rhythm. Coupling between nested prosodic units is identified as an additional source of rhythmic patterning in speech, and this coupling is claimed to be gradient and highly variable, dependent on speaker characteristics and text properties. Experimental results which illustrate several degrees of coupling between different prosodic levels are presented. A satisfactory account of speech rhythm will have to take both language-specific phonological properties and utterance-specific coupling among nested production units into account.

1 On Classification and Taxonomy

Taxonomy involves the determination of discrete classes. In its classical manifestation, living forms are divided into discrete groups (species, genera, families, etc), and criteria are established which help to decide which taxon a given exemplar should be assigned to. A basic assumption is that discrete classes exist underlyingly, and that a strict classification is, in principle, possible. In this regard it differs from the more general practice of biosystematics, which considers any and all relationships which exist among organisms.

There are many forms of linguistic taxonomy, most of which have the property that we have strong reason to suspect a discrete difference in some formal feature between the languages. For example, some languages have a basic word order in which the subject is ordered before the verb, which in turn precedes the object, while others order these three elements differently. Taxonomic licence is granted because of the discrete nature of the elements involved.

2 Prosody as a Basis for Taxonomy

Prosody has often been used as a basis for classifying languages. The grab bag of phenomena which can be linked under the label "prosody" leaves considerable scope for creative classification. Attempts have been made to classify languages based on stress, accent, intonation, lexical and morphological tone, and, of course, rhythm. However, it has not always been possible to unambiguously

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 28–35, 2002.

identify discrete elements corresponding to each of these dimensions with the same robustness as in the segmental, morphological or lexical domains.

Distinctions based on syllable structure have been fairly uncontroversial, as a segmental inventory is relatively easy to obtain for a given language, and the principles of syllable structure have shown considerable generality. Linguistic theories such as Autosegmental Phonology or Optimality Theory have provided well-founded and empirically supported theories of underlying discrete structures which permit classifications within and across languages.

Nowhere has the effort at establishing and defending a prosodic taxonomy had a harder time than in the domain of 'rhythm'. Without doubt, much of this lack of progress can be traced to differing interpretations of the term 'rhythm'. It will be a contention of this paper that at least two independent dimensions have been called to service in characterizing rhythm. One of these is related to syllable structure and segmental inventories, and may therefore offer the basis for a taxonomy. The other relates to a gradient phenomenon, not yet well understood, which mediates the role of syllables in determining macroscopic timing patterns. Its gradient nature precludes it from supporting a classification among languages. Furthermore, it will be claimed, pre-theoretical perceptions of rhythm (whether characteristic of a speaker or a language) are derived from an interplay between the discrete and the gradient phenomena.

3 Where Is Rhythm in Speech?

3.1 Rhythm across Languages

Our formal approaches to characterizing rhythm in speech are grounded in a pre-theoretical perception of a patterning in time which speech and music have, to some degree, in common. We become aware of something like rhythmic properties in speech when we contrast speech in different languages, and this is presumably the reason why rhythm has so-often been called upon to support language classification. The ability to distinguish among languages based on a signal which preserves low frequency information has been documented in infants [14], while Ramus demonstrated a similar ability in adults using resynthesized speech in which segments were stripped of their identity, but not their broad phonetic class [18]. Many attempts have been made to identify a basis for this apparent perception of a rhythmic difference among languages. Simplistic notions based on isochronous units have been uniformly rejected [6].

Two current influential models [19,10] take up a suggestion by Dauer [6] that languages may lie along a continuum (or in a continuous space), certain points of which have previously been identified with rhythmic classes (syllable-, stress- and mora-timed languages). They each develop continuous measures which can support clustering of languages in accordance with older taxonomic divisions. Since the introduction of the notion of gradient rhythmic qualities, it is no longer entirely clear that a taxonomy is being sought, as opposed to a more general systematic description of variation among languages.

3.2 Rhythm within Speaker

There is another, distinct, sense in which speech is rhythmical, and this is related to fluency. As we speak, the fluency with which speech is generated varies continually. We are all familiar with both the ease with which fluent speech flows, and the debilitating effect of its opposite, the dysfluent event. This type of rhythm is considerably harder to quantify, as it can vary substantially within a single utterance, and is apparently subject to the vagaries of expression and rhetorical force as much as to language-specific constraints.

This variability raises the question of whether the kind of index proposed by Ramus, Grabe and others can meaningfully be said to capture anything about *rhythm* in speech. The discrete basis for the suggested taxonomy can be argued to be grounded in segmental inventories and syllabic phonotactics, and can therefore be accounted for without reference to anything resembling the pre-theoretical notion of rhythm described at the start of this section. More succinctly, where is the bom-di-bom-bom in %V?

The argument to be developed here is that there are indeed two distinct phenomena here, which interact to provide a perception of rhythm in speech. On the one hand, there are linguistic units which vary discretely across languages. Thus English has its heavy and light syllables, stresses, feet etc, while Japanese has its Morae, perhaps a bi-moraic foot, and so on. These are symbolic, linguistic entities familiar from phonology, and language taxa can be constructed on foot[1] thereof. To some extent these alone dictate the alternation of light and heavy elements in spoken language, and so they contribute to the rhythmic signature of a language.

These units also serve as participants in hierarchical timing relationships, in which smaller prosodic units are nested within larger units, and the degree of coupling between levels varies in gradient fashion, as dictated by fluency, conversational intent, urgency, etc. As coupling varies continually, so too does the perceived rhythmicity of speech, and, perhaps, perceived fluency, though this direct association has yet to be tested.

The gradient coupling between prosodic levels (syllables within feet, feet within phrase, etc) has been identified and modelled before [16]. It has also been observed experimentally in the Speech Cycling paradigm [4,20], in which subjects repeat a short phrase in time with an external metronome. An example of the sort of simple pattern produced when repeating a short phrase is given in Fig. 1 (details in [4]). Each pattern corresponds to the nesting of a stress foot within the overall phrase repetition cycle. In related work, Tajima found evidence for the nesting of morae and perhaps a bi-moraic foot within the phrase repetition cycle [20].

4 Where Else to Look?

The work of Grabe and Ramus and colleagues [10,19] constitutes strong prima faciae evidence for categorical distinctions among languages based on the kind

[1] sorry.

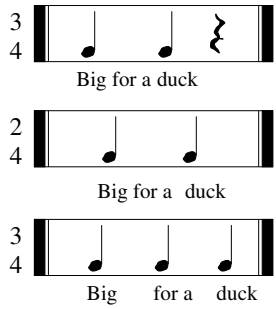


Fig. 1. Rhythmic patterns produced by English speakers in [4]

of linguistic unit on which rhythm is "hung". Evidence from Speech Cycling illustrates how, under rather extreme elicitation conditions, entrainment of one prosodic unit within another can be induced. Speech Cycling alone will not suffice to make the case that there is a continually varying level of entrainment between units at one level (syllables, perhaps feet) and prosodic units at a higher level (feet, perhaps phrases), as suggested by O'Dell and Nieminen [16] and Barbosa [2].

The claim being made here is that there is such entrainment, and that the degree of entrainment varies within speaker and across utterances. Because of this high degree of variability, the resulting rhythmic forms are not stable enough to support a rhythmic taxonomy. However, the sort of forms that can emerge are dictated largely by the discrete categories mentioned above, and so we will expect language-specific manifestations of entrainment between prosodic levels.

The evidence for temporal entrainment among prosodic units at distinct timescales under more natural speaking conditions is not uncontroversial. Attempts to identify compensatory shortening within the foot as unstressed syllables are added yielded negative results [13]. Some studies have produced weak evidence of compensatory durational adjustment toward weak isochrony [15,8], but most such investigations have been fruitless [6]. However, none of these investigations have considered the degree of entrainment between prosodic levels, and hence the strength of rhythmic regularity, to be a continuously variable function. We have recently found some intriguing evidence for a demonstrable entrainment between prosodic levels in read speech, without metronomic influence.

5 Metrical Structure

Methods. As part of a larger experiment still underway, speakers provided readings of word lists, where each list contained 8 trochaic forms (e.g. "tango, lighter, daddy, wiper, pony, cutter, pinky, mango"). A total of 54 readers each read 6 such lists in "as regular a form as possible". That is, they were instructed to produce something akin to an isochronous series. From each reading, P-centers, corresponding roughly to vowel onsets, were obtained by semi-automatic means

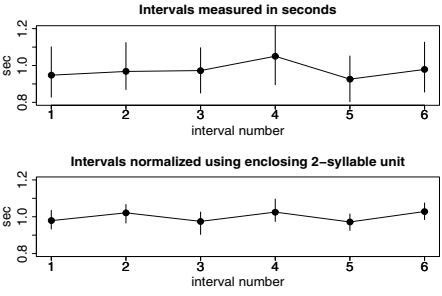


Fig. 2. Median and IQR of intervals from trochaic list reading task

(following the method of [4]), and the first six inter P-center intervals were plotted in several ways.

Results. Two plots are shown in Fig. 2. In the top panel, the first six inter-onset intervals have been computed, and each divided by the mean inter-onset interval. The median and IQR of each is shown (n=318), and the only interval which stands out is the fourth, separating the first group of four from the second. This interval is longer and more variable than all the others.

In the lower panel of Fig. 2, each interval has been normalized by a containing interval. For the first two intervals, the normalizing interval is the duration of the first two intervals, for intervals three and four, it is the sum of intervals three and four, and for five and six, it is the sum of intervals five and six. In order to make these measurements directly comparable with those of the top panel, all normalized intervals are again divided by the mean for the whole data set. This representation of interval duration tells a very different story. Now interval duration, expressed as a proportion of a containing two-interval unit, is much less variable. There is also a clear alternating pattern, where the first interval of each two-interval "foot" is shorter than the second.

Interval durations expressed in milliseconds are highly variable, reflecting rate variation across list readings and from one speaker to the next. When each interval is re-expressed as a proportion of a containing interval, however, the data become much more coherent.

The task of reading a regular list of 8 trochees, while not as rhythmically constrained as speech cycling, is still carefully designed to elicit maximally rhythmical speech production. Given speech material which lends itself to simple rhythmical grouping, speakers do indeed impose a rhythmic organization on their speech, resulting in durations which are interpretable in terms of simple meter. Not all speech is this regular, however. In the following section, we report some new data which provides tentative support for the hypothesis that hierarchical timing is imposed under much less stringent speaking conditions.

6 Temporal Structure as Characteristic of an Individual Speaker

Methods. In the course of a larger experiment, readings from 27 speaker pairs were obtained reading the first paragraph of the rainbow text. For each pair of speakers, A and B, a reading was first obtained from A, then A and B read together, attempting to remain in synchrony with one another, then Speaker B read the text. After some intervening practice at this, the process was repeated, with Speaker B starting, then A and B together, and finally Speaker A. From each recording, the final sentence ("When a man looks for something beyond his reach, his friends say he is looking for the pot of gold at the end of the rainbow") was excised, and 16 well defined points in the waveform were identified by hand. These points correspond to reliably recognizable events such as stop releases, vowel onsets etc, and together they divided the utterance into 15 sub-intervals of approximately 2–4 syllables each.

Results. This sequence of 15 intervals can again be viewed in two ways. Firstly, we can consider the vector of 15 millisecond values, each expressing a well defined interval. We would naturally expect two utterances recorded in the synchronous condition to be fairly similar by this measure.

However, we can obtain a very crude representation of the rhythmical structure of an utterance by expressing each interval instead as a proportion of some larger containing interval. The above sentence is normally read as two intonational phrases (separated at the comma), so we can re-express the sequence of measurements such that each interval is now given as a proportion of the containing IP (or the measurement points most nearly located at the two ends of that IP). This is also a vector of intervals, but each is expressed as a function of the overall temporal organization of the phrase.

Something rather surprising happens when we consider the similarity of two utterances using these two measures. Using absolute durations, we found that two utterances produced by two speakers speaking in synchrony appeared similar. No surprise there. When we normalized the durations using the containing IP length, the apparant similarity vanished completely. If we instead compared utterances produced by the same speaker, once reading alone and a second time reading in synchrony with another speaker, absolute duration measurements showed there to be some, but not much similarity between utterances. When we normalized these durations, however, the similarity between utterences produced by the same speaker was very strong indeed, despite the great difference in elicitation circumstances. Details are given in [5].

7 Discussion

Both the preceding experimental results illustrate the coordination of temporal intervals at one level with those at a higher level. In the word list example, metrical structure based on the hierarchical nesting of each word within a

two-word unit was evident. In the preceding example, a sequence of temporal intervals in which each interval is expressed as a proportion of a larger interval was demonstrated to be characteristic of an individual speaker, and quite stable across different elicitation conditions. This accords with the finding that timing at both phoneme and word level remains largely unaltered in speech produced by professional mimics, even though the resulting speech is perceived to be similar to the target voice [7,21].

All of which brings us back to the subject of speech rhythm. The argument was made that a gradient phenomenon, not yet well understood, mediates the role of syllables in determining macroscopic timing patterns. Its gradient nature precludes it from supporting a classification among languages. Furthermore, it was claimed, pre-theoretical perceptions of rhythm (whether characteristic of a speaker or a language) are derived from an interplay between the discrete and the gradient phenomena. The intervals between stressed syllable onsets have long been held to be of singular importance in the perception of English speech rhythm.

In the word list experiment, we saw that these intervals do in fact partake in a strictly metrical structure, demonstrable and measurable in real time, when the spoken material is sufficiently regular. The units (feet delimited by stressed syllables) are language specific (Japanese, for example, has no correlate of stress), but the participation of these units in genuinely rhythmical structures is dependent on the nature of the spoken utterance.

In the second experiment we saw that the entrainment among levels does exist in some form when the material is less regular. The resulting pattern is not perceived as being rhythmic in a musical sense, but in common with the simple metrical example, there is a demonstrable coupling between intervals at one prosodic level and those at a higher level.

Little is known about the nature or origin of these production constraints which impose hierarchical temporal structure upon an utterance. The similarity which can be observed between speech cycling patterns and patterns of coordination among the limbs [3] suggests that the origin is to be sought in the demands imposed by the finely tuned coordination of heterogeneous components in speech production, and is thus one aspect of motor control in speech. But the elements upon which these patterns are built are embedded in the phonological regularities which typify a given language. Progress in the study of speech rhythm will require taking both the linguistic units and their forms of coordination into account.

Acknowledgments. Keiichi Tajima (ATR) helped in preparation of the word lists. Work supported by a grant from the Irish Higher Education Authority.

References

1. David Abercrombie. *Elements of general phonetics.* Aldine Pub. Co., Chicago, IL, 1967.

2. Plínio Almeida Barbosa. Explaining cross-linguistic rhythmic variability via a coupled-oscillator model of rhythm production. In *Proceedings of Prosody 2002*. 2002. to appear.
3. Fred Cummins and Robert F. Port. Rhythmic commonalities between hand gestures and speech. In *Proceedings of the Eighteenth Meeting of the Cognitive Science Society*, pages 415–419. Lawrence Erlbaum Associates, 1996.
4. Fred Cummins and Robert F. Port. Rhythmic constraints on stress timing in English. *Journal of Phonetics*, 26(2):145–171, 1998.
5. Fred Cummins. Measuring the temporal distance between two utterances. In *Proceedings of ICSLP 2002*, Denver, CO., 2002. submitted.
6. R. M. Dauer. Stress-timing and syllable-timing reanalyzed. *Journal of Phonetics*, 11:51–62, 1983.
7. Anders Eriksson and Pär Wretling. How flexible is the human voice?–a case study of mimicry. In *Proceedings of EUROSPEECH*, volume 2, pages 1043–1046, Rhodes, Greece, 1997.
8. Edda Farnetani and Shiro Kori. Effects of syllable and word structure on segmental durations in spoken Italian. *Speech Communication*, 5:17–34, 1986.
9. A. Galves, J. Garcia, D. Duarte, and C. Galves. Sonority as a basis for rhythmic class discrimination. In *Proceedings of Prosody 2002*. 2002. to appear.
10. Esther Grabe and Ee Ling Low. Durational variability in speech and the rhythm class hypothesis. In *Papers in Laboratory Phonology 7*. 2000. to appear.
11. Carlos Gussenhoven. Discreteness and gradience in intonational contrasts. *Language and Speech*, 42(2–3):283–305, 1999.
12. D. Robert Ladd and Rachel Morton. The perception of intonational emphasis: continuous or categorical? *Journal of Phonetics*, 25:313–342, 1997.
13. Lloyd H. Nakatani, Kathleen D. O'Connor, and Carletta H. Aston. Prosodic aspects of American English speech rhythm. *Phonetica*, 38:84–106, 1981.
14. T. Nazzi, J. Bertoncini, and J. Mehler. Language discrimination by newborns: towards an understanding of the role of rhythm. *Journal of Experimental Psychology: Human Perception and Performance*, 24:756–766, 1998.
15. Sieb G. Nooteboom. *Production and Perception of Vowel Duration*. PhD thesis, Utrecht, The Netherlands, 1972.
16. Michael L. O'Dell and Tommi Nieminen. Coupled oscillator model of speech rhythm. In *Proceedings of the International Congress of Phonetic Sciences*, San Francisco, 1999.
17. Janet B. Pierrehumbert. *The Phonology and Phonetics of English Intonation*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1980. Reprinted by the Indiana University Linguistics Club.
18. Franck Ramus and Jacques Mehler. Language identification with suprasegmental cues: A study based on speech resynthesis. *Journal of the Acoustical Society of America*, 105(1):512–521, 1999.
19. Franck Ramus, Marina Nespor, and Jacques Mehler. Correlates of linguistic rhythm in the speech signal. *Cognition*, 73(3):265–292, 1999.
20. Keiichi Tajima. *Speech Rhythm in English and Japanese: Experiments in Speech Cycling*. PhD thesis, Indiana University, Bloomington, IN, 1998.
21. Pär Wretling and Anders Eriksson. Is articulatory timing speaker specific? – evidence from imitated voices. In *Proc. FONETIK 98*, pages 48–52, 1998.

Finding Agents in a Two-Dimensional Boolean STaM

Jim Doran

Department of Computer Science
University of Essex
Colchester, UK, CO4 3SL
doraj@essex.ac.uk

Abstract. A simple discrete mathematical model of a space, time and matter manifold, called a *STaM*, is described which is intended as a framework within which to study agents and multi-agent systems. In this paper we concentrate on two-dimensional Boolean STaMs, and report implemented algorithms for detecting agents within them. A specific example STaM is given, some experimental results presented, and conclusions drawn.

1 Introduction

In a previous paper [3] we have proposed a simple, discrete model of a space, time and matter manifold that we call a "STaM" (for "Space, Time and Matter"). The proposal embodies three main ideas. The first is to seek to understand agents (including "intelligent" agents) and multiple agent systems in terms of their underlying *material structure* and its properties. The approach is therefore strongly reductionist. Its advantages are that it is natural and that it enables a rich array of questions to be addressed in an integrated way. Secondly, we employ the concept of *"block" space-time*. Familiar in space-time physics, this concept has rarely seemed relevant in artificial intelligence work. However, in the model to be presented we seek an objective perspective, and "block-time" avoids the anthropocentrism inherent in the notion of "now", with its connotation of a fixed past and an open future. Finally, we seek not to design agents, but to *recognise* them. This opens a door to aspects of agency that would not be encountered using the current repertoire of agent design techniques. For example, it makes it easier to recognise possibilities intermediate between conceptually distinct extremes.

As will appear, a STaM is similar to the trace of a Boolean or multi-state network or of a cellular automaton. However we shall focus attention on the trace rather than on the rule system that generates it, and interpret the trace in an unusual way.

1.1 STaMs

We define a STaM to be an N-dimensional finitely bounded integer vector space, S, with a mapping from its elements to elements of a set M. For our purposes it is sufficient to assume a suitable inner product defined over S so that it is (at least) locally

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 36-44, 2002.

Euclidean. The elements of the set M correspond notionally to a finite enumeration of different possible states of matter.[1]

Intuitively a STaM is a space-time region, with (discrete) locations identified by integer N-tuples (Cartesian coordinates), and with a matter state associated with each location. In computational terms, a STaM could be a regular multi-dimensional array of fixed values over an arbitrary enumerated type (the matter states). One dimension of the STaM is naturally taken to be temporal and directed, past to future, and the remaining dimensions to be spatial. This straightforwardly gives a temporal sequence of spatial *STaM states*. Thus an intuitive example of a STaM is a 4D rectangular cuboid or "block", with one dimension interpreted as time and the remaining three as Euclidean space. We shall refer to the *locations*[2] of the STaM and their *(location-) states (of matter)*. Note also that by a *spatial location* we shall mean not one specific location in the STaM, but rather a temporal sequence of successive locations with the same spatial coordinates. For brevity we shall usually refer to a spatial location as an *S-location*.

In this paper we consider only STaMs that are two dimensional (so that a spatial state is one dimensional) and Boolean in the sense that there are only two alternative matter states. Furthermore, we shall "join up" the edges of the STaM so that it becomes a torus.

2 Regularities and Rules in STaMs

A particular STaM may or may not have regularities within it, and any regularities there are may be of many different kinds. We assume that regularities, however discovered, are to be expressed in the form of *condition* ➔ *consequence* rules that specify the matter state of a particular location by reference to certain earlier (i.e. temporally preceding) location states, without use of spatial variables. We allow conjunctions on the left hand side. The set of locations referenced by the LHS of a rule we call the rule's *neighbourhood*. A neighbourhood need not necessarily be spatially contiguous (in terms of the spatial structure of the STaM). Rule sets may or may not be homogeneous in space, that is, the rules may or may not be the same for all S-locations, and may or may not be Markovian in the sense that the LHS and RHS of the rules refer only to successive STaM states. Thus these rules somewhat generalise the update rules of a Boolean or multi-state network.

The regularities in a particular STaM may be captured by many alternative rule sets, and each may be regarded as an *interpretation* of the STaM. Different interpretations will have different properties.

2.1 Influence and Influence Elements

We shall say that one S-location X *influences* another Y, with respect to a particular rule set, if there exists a rule R within the rule set that determines a state of Y wholly

[1] A STaM can be seen as a special and very simple case of the space-time manifold, with a distribution of matter-energy-momentum over it, that lies at the heart of general relativity theory (see, for example, [4]).

[2] We avoid the commonly used term "event" as liable to confuse in this context.

or partially by reference to the state of X (that is, the LHS of R references the state of X and the RHS of R specifies the state of Y). A particular instance of a rule with location states discarded we call an *influence element.*

2.2 What Are STaM Rules?

A rule set associated with a STaM is somewhat analogous to physical laws in that it determines the dynamics within the STaM. Although this analogy may seem remote, it does guide our intuition in a useful direction. However, it is important to realise that, in the perspective of an outside observer, the rules merely *predict* their consequence state(s). They do not generate them or otherwise bring them into being for, by assumption, the entire STaM already exists as a prior structure.

3 Agents in STaMs

Given the notion of a STaM and its associated rule sets, we now ask if there can be sets of locations and/or states within a STaM that may reasonably be regarded as agents. Of course, and crucially, this requires us to decide just what "an agent" is. The agent definitions found in the literature of agent technology typically refer to such complex and ambiguous notions as "autonomy", "pro-activity", "communication", and "sociability", and thus offer rather little help in this context. However, the standard AI textbook by Russell and Norvig [6, p. 31] describes an agent a little more usefully as "anything that can be viewed as perceiving its environment through sensors, and acting upon that environment through effectors, and Weiss [7, p. 584] gives as one definition of an agent "an active object or a bounded process with the ability to perceive, reason and act". At least as imprecise, is Price's remark that when seeing an agent "an Augustinian god just sees a nexus for a complicated structure of correlations" [5, p.169]. However, this does give us the important idea of an agent as some kind of locus of complexity in space-time.

3.1 A-Agents

We focus upon a certain type of agent, which we call an *A-agent*, with the following definition.

Definition: With respect to a particular rule set, R, associated with a STaM, an *A-agent* is a fixed non-empty set of S-locations, each S-location extended over the same non-empty time interval, the A-agent's *temporal extension*. Two disjoint non-empty subsets of the agent's S-locations are designated *input* and *output* S-locations. The remaining S-locations of the agent are *internal*. Input, output and internal S-locations have the following properties:

- An *input* S-location is influenced only by S-locations external to the agent and influences only output S-locations and/or S-locations internal to the agent.
- An *output* S-location is influenced only by input S-locations and/or internal S-locations of the agent and influences only S-locations external to the agent.
- An *internal* S-location is influenced by, or influences, only other S-locations of the agent i.e. internal, input or output S-locations.

Notice that this definition of an A-agent rests directly on the definition of what it means for one S-location to influence another, and therefore rests in turn on a set of rules associated with the STaM and their particular influence elements.

The S-locations that comprise an A-agent are fixed over time and need not necessarily be spatially contiguous. Not any set of S-locations is an A-agent. In particular, a single S-location cannot be an A-agent (the input and output sets must be non-empty and disjoint) and a pair of locations (L1, L2) is an A-agent, with L1 as input and L2 as output, if and only if: L1 influences at most L2, L2 does not influence L1, and L2 is only influenced by L1

There may, of course, be many A-agents in a given STaM and there is nothing in the definition that prevents there being A-agents that overlap, or are nested one within the other. Our definition of an agent permits some interesting special cases. For example, if a STaM is toroidal, and therefore temporally finite but unbounded, then clearly an A-agent within it may also be.

3.2 The Capabilities of A-Agents

An A-agent meets some of the most often cited requirements of an agent [6,7]: it "senses" (via its input S-location(s)), "decides" (via its internal S-location(s)), and "acts" (via its output S-location(s)). A-agents are, in effect, traces of (recurrent) multi-state networks [1]. They have state (the collective states of their internal S-locations), and, in effect, use "if-then" rules to select actions. They can adapt and learn by updating their state. Thus the internal processing and complexity of A-agents should not be underestimated. It is akin to that of many of the agents discussed in the literature, and can even include elements of high-level "cognition".

4 An Experiment with a Specific STaM

It is likely that a randomly generated STaM, although easy to create, will rarely contain interesting agents. Therefore a more complex computational experiment has been conducted according to the following design:

1. Generate a STaM with known and non-trivial A-agents within it
2. Fit rules to the generated STaM
3. Use the rules to identify A-agents in the STaM

In addition to testing the viability of this three-stage process, the experimental objective was to discover whether the A-agents "build into" the STaM in Stage 1, would be recovered in Stage 3 or whether a different set would be found.

5 The MICRO-COUNT and MICRO-SETUP Problems

We consider a certain type of problem, the solutions to which can stand as agents in a STaM. Suppose that a set of S-locations is given and that two of them are specified as input and output S-locations. The successive states of the input and output S-locations are specified. The requirement is then to find a rule set that so determines the succes-

sive values of the remaining S-locations that the imposed "boundary conditions" are met. The boundary conditions comprise the given input and output state sequences, together possibly with given sets of initial and terminal location states at the commencement and end of the relevant temporal extension.

We assume Markovian but spatially heterogeneous rules, whose neighbourhoods are constrained to be of a specified size and are the same for all the states of a particular S-location.

We call a problem of this type a *sensory action Boolean network* (SABN) problem, and refer to the task of meeting the boundary conditions as that of *supporting the history*.

5.1 The MICRO-COUNT Problem

The COUNT problem is an example of a SABN problem (see [3]). It requires an agent to "count" how many '1's there are in each of a set of strings of '1's and to respond accordingly. The sensory input is (conceptually) structured as a series of episodes. In each episode a different '1' string of length < 8 is presented to the agent. During an episode the network agent must "count" the length of the '1' string presented to it and return as its corresponding "action" output string four bits in which the first bit is always '1' and the following three bits a binary encoding of the length of the string. For example, the sensory input '11111' must lead to the action output '1101'. In the (even!) simpler MICRO-COUNT problem considered here, just three episodes are presented, containing the string lengths 1 to 3, and only 1+2 bits are used to encode each answer. The entire input and output sequences (each of length 45) for the MICRO-COUNT problem are given in Table 1.

Table 1. The sensory input (S) and action output (A) sequences for the MICRO-COUNT Problem. There are three episodes as shown. These episodes, however, are not distinguished in the input and output sequences as presented to the agent

S	100000000000000	110000000000000	111000000000000
A	000000101000000	000000110000000	000000111000000

5.2 The MICRO-SETUP Problem

The MICRO-SETUP problem is similar to the MICRO-COUNT problem and is in a sense its inverse. Given the output from the MICRO-COUNT agent, the MICRO-SETUP agent "sets up" the next episode for the MICRO-COUNT agent. The specification of the MICRO-SETUP problem is given in Table 2.

Table 2. The sensory input (S) and action output (A) sequences for the MICRO-SETUP Problem. Again there are three episodes, but note their displacement with respect to the episodes of MICRO-COUNT

S	00000000	101000000000000	110000000000000	1110000
A	00000000	000000110000000	000000111000000	0000001

5.3 Solving SABN Problems

To solve a SABN problem, a consistent rule set must be found to update each S-location. Recall that the rules fitted are to be Markovian, but may be heterogeneous. The simplest way to obtain a suitable set of rules is by systematic enumeration and test. Of course, this is impossibly slow. We have therefore implemented (in the C language) a program that uses an *ad hoc* version of hill-climbing search with random restart and that obtains solutions to SABN problems relatively quickly on a fast PC.

We have used our program to obtain solutions for both the MICRO-COUNT and MICRO-SETUP problems. Both solutions use 15 S-locations (including the sensory and action S-locations) with rule neighbourhoods of size 5.

5.4 Interpreting the Solutions

Finding solutions to SABN problems such as MICRO-COUNT is one thing, discovering just how a solution actually supports the corresponding history is quite another.[3] Is it purely "reactive", so that each input pattern is directly linked to its required output, or is there some definite process of counting within the solution? Or is there some "noisy" mixture of the two? Or something else? In a trivial sense what happens is clear. Rules "fire" successively and the required outcome is achieved. But what conceptual repertoire might support a more insightful analyse of the internal dynamics of an agent of this kind? Dynamic systems theory is an established candidate [2,8], with analyses typically couched in terms of different types of attractor and of trajectories to and from them. An arguably more attractive possibility is, where appropriate, to find interpretations of the solutions in terms of higher-level cognitive processes such as predictive planning. This possibility is discussed in [3].

6 An Example STaM

Table 3 presents (part of) a STaM constructed from the solution traces obtained for the MICRO-COUNT and MICRO-RESET problems, interpretable (by reference to the associated rule sets) as A-agents, with the addition of two S-locations interpretable as the shared environment of the agents. Although this STaM has been constructed by hand, it could have been constructed in some other quite different way, for example by random generation. Thus the STaM in Table 3 is not necessarily tied to the two SABN problems. But, of course, we know that it *can* bear the interpretation of containing two A-agents that are engaged in a problem solving and problem setting interaction. We now investigate what *other* interpretations of this STaM are possible.

[3] Many authors have commented upon this difficulty, e.g. " … it is common to achieve a perfectly competent network whose operation appears to be completely incomprehensible." [2, p. 470].

7 Fitting Rules and Finding A-Agents in the Example STaM

The task of discovering A-agents in the example STaM (or in any other STaM) falls into two stages: fitting rules to express regularities in the STaM, and then using the fitted rules to identify actual agents.

We have implemented, again in C, a program that seeks to associate with every location state in the STaM a combination of at most three location states at the immediately preceding time that predict it without error. Each successful association is an instance of a condition-consequence rule. Recall that an *influence element* is a rule instance from which location states have been discarded, and therefore has the form: <A,t> & <B,t> & <C,t> ➔ <D,t+1> where A,B,C,D name specific S- locations and t is a variable over times. A count is kept of the number of occurrences of each influence element in the STaM.

Table 3. Part of a STaM composed from solutions to the MICRO-COUNT and MICRO-SETUP problems (rows 1-15 and rows 17-31 respectively), with additional "environmental" rows 16 and 32. An interpretation is that the agent MICRO-SETUP sends problems to the agent MICRO-COUNT across the environment, and that MICRO-COUNT solves them and passes the answers back to MICRO-SETUP again across the environment. As the STaM is toroidal, both agents are temporally unbounded and exist in a finite circle of time of length 45

1	100000000000000110000000000000111000000000000
2	000110111101101101110011101111100100110010110
3	110000000100000011101101110100100001110000010
4	010011100000000100001101000000001111110101110
5	011101011011010100110000101010110100110010010
6	110010111110000101101001010100101011101100100
7	100101011000001011110000000000001111010001000
8	001011010110111101111110000010101000001000010
9	101001000110001111101100000001100100010100100
10	111110111110100011111110100010111010101001100
11	011101011111010000011111010110011100000100110
12	101000010000000001100011111010101001110110110
13	010101100011010111010111101000110100101110010
14	100101111011010000000111011101000011110101100
15	000000101000000000000110000000000000111000000
16	000000010100000000000011000000000000011100000
17	000000001010000000000001100000000000001110000
18	…
19	…
20	…
.	
.	
31	000000000000001100000000000001110000000000001
32	000……..

The list of influence elements derived from the STaM is passed to a further program, again implemented in C, which finds A-agents. This program first discards

influence elements whose occurrence count is below some threshold. In effect it ignores "weak" influences. The program then tests all directed pairs of S-locations, recording those pairs that can stand as input and output S-locations for A-agents as defined earlier. Only singleton input and output sets are considered, and that only those A-agents whose temporal extension is unbounded are sought.

7.1 Results

The total number of location states in the STaM given in Table 3 is 1440 (32x45). The rule finding program finds 1170 rules that together cover all location states, and a total of 680 influence elements with counts varying from 1 to 45.

The agent-finding program, discarding elements with count below 10, finds *24 A-agents*: {1,27}, {1,28}, {1,32}, {2,27}, {2,28}, {2,32}, {5,13}, {7,21}, {9,13}, {15,16}, {15,17}, {16,17}, {23,27}, {23,28}, {23,32}, {24,21}, {25,1}, {25,27}, {25,28}, {25,32}, {31,1}, {31,27}, {31,28}, {31,32}, where each directed pair gives the input and output S-locations respectively of an A-agent. The original MICRO-COUNT and MICRO-SETUP A-agents are *not* found, but other "obvious" A-agents are, for example the A-agent {15,17} that comprises the three S-locations 15,16,17, and draws one S-location from each of the two original agents and one from their environment. Notice that the A-agents that are found often share S-locations.

The primary reason why the original A-agents are not found is that they were created using rules with neighbourhoods of size 5, whereas the relatively simple rule finding program employed in the experiment can seek only rules with neighbourhoods of size at most three.

8 Discussion and Conclusion

The work reported here offers a precise approach to the study of agents and multi-agent systems that is related to, but different from, the neural network, Boolean network and cellular automata traditions. The notion of a STaM grounds the concept of an agent in a way that permits a range of interesting and interdisciplinary questions to be asked. It may seem counter-intuitive to interpret a fixed array of ones and zeroes as an agent, but the work presented here demonstrates that it is quite meaningful to do so. A key unanswered question is how to determine algorithmically whether or not an A-agent embodies any element of higher-level "cognition".

The main conclusion of the experiment reported is that a STaM can be interpreted in more than one way. This is because descriptive rules can be fitted to a STaM in a many ways, depending upon the exact design of the fitting algorithm and upon what are regarded as desirable properties of a rule set. Different sets of rules then determine different set of agents within the STaM. To the extent that a STaM is accepted as a useful model of space-time as we experience it, then the implications for our view of agents in the world around us are thought provoking.

References

1. Angeline, P. J., Saunders, G. M., Pollack, J. B.: An Evolutionary Algorithm that Constructs Recurrent Neural Networks. IEEE Transactions on Neural Networks, Vol. 5, No. 1 (1994) 54–65
2. Beer, R.: On the Dynamics of Small Continuous-Time Neural Networks. Adaptive Behavior, Vol. 3, No. 4, (1995) 469–509
3. Doran, J. E.: Agents and MAS in STaMs. In: Foundations and Applications of Multi-Agent Systems: UKMAS 1996–2000 (ed M. d'Inverno, M. Fisher, M. Luck, C. Preist), Springer Verlag (in press).
4. Naber, G. L.: Spacetime and Singularities: An Introduction. Cambridge University Press, Cambridge (1988)
5. Price, H.: Time's Arrow & Archimedes' Point. Oxford University Press, Oxford and New York (1996)
6. Russell, S., and Norvig, P. (eds.): Artificial Intelligence: a Modern Approach. Prentice Hall (1995)
7. Weiss, G. (ed.): Multiagent Systems. The MIT Press, Cambridge, Mass. and London, England (1999)
8. Wuensche, A.: Discrete Dynamical Networks and their Attractor Basins. Complexity International, Volume 6, (online), and SFI Working Paper, 98-11-101 (1998)

Neuro-symbolic System for Forecasting Red Tides*

Florentino Fdez-Riverola[1], Juan M. Corchado[2], and Jesús M. Torres[3]

[1] Dpto. de Informática, E.S.E.I., University of Vigo
Campus Universitario As Lagoas s/n., 32004, Ourense, Spain
riverola@uvigo.es
[2] Dpto. de Informática y Automática, University of Salamanca
Facultad de Ciencias, Plaza de la Merced, s/n., 37008, Salamanca, Spain
corchado@usal.es
[3] Dpto. de Física Aplicada, University of Vigo
Facultad de Ciencias, Lagoas Marcosende, 36200, Vigo, Spain
jesu@uvigo.es

Abstract. A hybrid neuro-symbolic problem solving model is presented in which the aim is to forecast parameters of a complex and dynamic environment in an unsupervised way. In situations in which the rules that determine a system are unknown, the prediction of the parameter values that determine the characteristic behaviour of the system can be a problematic task. In such a situation, it has been found that a hybrid case-based reasoning (CBR) system can provide a more effective means of performing such predictions than other connectionist or symbolic techniques. The system employs a CBR model to wrap a growing cell structures network, a radial basis function network and a set of Sugeno fuzzy models to provide an accurate prediction. Each of these techniques is used in a different stage of the reasoning cycle of the CBR system to retrieve historical data, to adapt it to the present problem and to review the proposed solution. The results obtained from experiments, in which the system operated in a real environment, are presented.

1 Introduction

Forecasting the behaviour of a dynamic system is, in general, a difficult task, especially if the prediction needs to be achieved in real time. In such a situation one strategy is to create an adaptive system which possesses the flexibility to behave in different ways depending on the state of the environment. This paper presents the application of a novel hybrid artificial intelligence (AI) model to a forecasting problem over a complex and dynamic environment. The approach presented is capable of producing satisfactory results in situations in which neither artificial neural network nor statistical models have been sufficiently successful.

* This research was supported in part by PGIDT00MAR30104PR project of Xunta de Galicia, Spain

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 45–52, 2002.

The oceans of the world form a highly dynamic system for which it is difficult to create mathematical models [1]. *Red tides* are the name for the discolourations caused by dense concentrations of the microscopic plants of the sea, the so-called phytoplankton. The discolouration varies with the species of phytoplankton, its pigments, size and concentration, the time of day, the angle of the sun and other factors. Red tides usually occur along the north west of the Iberian Peninsula in late summer and autumn [2]. The rapid increase in dinoflagellate numbers, sometimes to millions of cells per liter of water, is what is known as a *bloom* of phytoplankton (if the concentration ascends above the 100.000 cells per liter). The type of dinoflagellate on which we focus in this study is the pseudo-nitzschia spp diatom which is known to cause amnesic shellfish poisoning (ASP).

An AI approach to the problem of forecasting in the ocean environment offers potential advantages over alternative approaches, because it is able to deal with uncertain, incomplete and even inconsistent data. Several types of standard artificial neural network (ANN) have been used to forecast the evolution of different oceanographic parameters [3,4,5]. Our aim is to develop an autonomous and reliable forecasting mechanism. The results obtained to date suggest that the approach to be described in this paper appears to fulfil this aim.

The work presented in this paper is based on the successful results obtained with the hybrid CBR system reported in [4,5,6] and used to predict the evolution of the temperature of the water ahead of an ongoing vessel, in real time. The hybrid system proposed in this paper is an extension and an improvement of the previously mentioned research. The retrieval, reuse, revision and learning stages of the CBR system have been modified or changed for two reasons: to adapt the hybrid system to the previously mentioned problem and to automate completely the reasoning process of the proposed forecasting mechanism.

The structure of the paper is as follows. First a brief overview of the CBR systems for forecasting is presented. Then the red tide problem domain is briefly outlined. The hybrid neuro-symbolic system is then explained, and finally, the results obtained to date with the proposed forecasting system are presented and analyzed.

2 CBR Systems for Forecasting

Several researchers [7,8], have used k-nearest-neighbour algorithms for time series predictions. Although a k-nearest-neighbour algorithm does not, in itself, constitute a CBR system, it may be regarded as a very basic and limited form of CBR operation in numerical domains. Other examples of CBR systems that carry out predictions can be found in [9,10,11,12,13].

In most cases, the CBR systems used in forecasting problems have flat memories with simple data representation structures using k-nearest-neighbour metrics in their retrieve phase. K-nearest-neighbour metric are acceptable if the system is relatively stable and well understood, but if the system is dynamic and the forecast is required in real time, it may not be possible to easily redefine the k-nearest-neighbour metrics adequately. The dominant characteristic of

the adaptation stage used in these models are similarity metrics or statistical models, although, in some systems, case adaptation is accomplished manually. If the problem is very complex, there may not be an adaptation strategy and the most similar case is used directly, but it is believed that adequate adaptation is one of the keys to a successful CBR paradigm. In the majority of the systems surveyed, case revision (if carried out at all) is performed by human expert, and in all the cases the CBR systems are provided with a small case-base. A survey of such forecasting CBR systems can be found in [14]. In this paper a method for automating the CBR reasoning process is presented for the solution of problems in which the cases are characterised predominantly by numerical information.

Traditionally, CBR systems have been combined with other technologies such as artificial neural networks, rule-based systems, constraint satisfaction problems and others, producing successful results [15]. Our proposal requires to embed two artificial neural networks and a set of fuzzy systems in the CBR life cycle.

3 Forecasting Red Tides

In the current work the aim is to develop a system for forecasting the concentrations of the pseudo-nitzschia spp, that it is the diatom that produces the most harmful red tides, at different geographical points one week in advance.

The problem of forecasting, which is currently being addressed, may be simply stated as follows:

- **Given**: a sequence of data values (representing the current state and the immediately previous one) about some physical and biological parameters,
- **Predict**: the value of a parameter at some future point(s) or time(s).

In order to forecast the concentration of pseudo-nitzschia spp at a given point one week in advance, a problem descriptor is generated on a weekly basis. A problem descriptor consists of a sequence of N sampled data values (filtered and preprocessed) recorded from the water mass for which the forecast is required, and the collection time and date. Every week the concentration of pseudo-nitzschia spp is added to a problem descriptor forming a new input vector. The problem descriptor is composed of a vector with the variables that characterise the problem recorded during two weeks. The prediction or output of the system is the concentration of pseudo-nitzschia spp one week after, as indicated in Table 1.

The forecasted values are obtained using a neural network enhanced hybrid case-base reasoning system. Figure 1 illustrates the relationships between the processes and components of the hybrid CBR system. The cyclic CBR process shown in Fig. 1 the has been inspired by the work of [4] and [5]. The diagram shows the technology used in each stage, where the four basic phases of the CBR cycle are shown as rectangles.

The retrieval stage is carried out using a Growing Cell Structures (GCS) ANN [16]. The GCS facilitates the indexing of cases and the selection of those that are more similar to the problem descriptor. The GCS network groups similar cases and forms classes. When a new problem is presented to this network, it

Table 1. Variables that define a case

Variable	Unit	Week
Date	dd-mm-yyyy	W_{n-1}, W_n
Temperature	Cent. degrees	W_{n-1}, W_n
Oxygen	milliliters/liter	W_{n-1}, W_n
PH	acid/based	W_{n-1}, W_n
Transmitance	%	W_{n-1}, W_n
Fluorescence	%	W_{n-1}, W_n
Cloud index	%	W_{n-1}, W_n
Recount of diatoms	cel/liter	W_{n-1}, W_n
pseudo-nitzschia spp	cel/liter	W_{n-1}, W_n
pseudo-nitzschia spp (future)	*cel/liter*	W_{n+1}

is associated to its most representative class and all members of such class are retrieved. The reuse of cases is carried out with a Radial Basis Function (RBF) ANN [17], which generates an initial solution creating a model with the retrieved cases. The GCS network guarantees that these cases are homogeneous and can be modeled by the RBF network. The revision is carried out using a group of pondered Fuzzy systems that identify potential incorrect solutions. Finally, the learning stage is carried out when the real value of the concentration of pseudo-nitzschia spp is measured and the error value is calculated, updating the knowledge structure of all the system. The cycle of operations of the hybrid system is explained in detail in Section 3.1.

3.1 System Operation

The forecasting system uses data from two main sources: The raw data (sea temperature, salinity, PH, oxygen and other physical characteristics of the water mass) which are weekly measured by the monitoring net of toxic proliferations in the CCCMM (Centro de Control da Calidade do Medio Marino, *Oceanographic environment Quality Control Centre*, Vigo, Spain), consist of a vector of discrete sampled values (at 5, 10 and 15 meters deep) of each oceanographic parameter used in this experiment, in the form of a time series. These data values are complemented by additional data derived from satellite images, which are received and processed daily, and other data belonging to ocean buoys that record data on a daily bases. Table 1 shows the variables that characterise the problem. Data of the last 2 weeks (W_{n-1}, W_n) is used to forecast the concentration of pseudo-nitzschia spp one week ahead (W_{n+1}).

The cycle of forecasting operations (which is repeated every week) proceeds as follows. When a new problem is presented to the system, the GCS neuronal network is used to obtain the k most similar cases to the given problem (identifying the class to which the problem belongs).

In the reuse phase, the values of the weights and centers of the neural network [17] used in the previous forecast are retrieved from the knowledge base. These

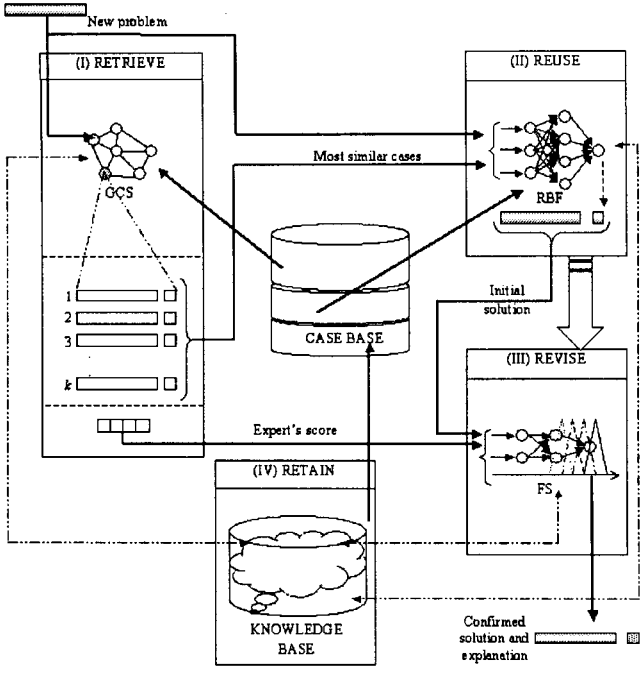


Fig. 1. Hybrid neuro-symbolic system

network parameters together with the k retrieved cases, are then used to retrain the RBF network and to obtain an initial forecast of the concentration of pseudo-nitzschia spp. During this process the values of the parameters that characterise the network are updated.

In the revision phase, the initial solution proposed by the RBF neural network is modified according to the responses of the four Fuzzy revision subsystems. Each revision subsystem has been created from the RBF network using neurofuzzy techniques [18]. For each class of the GCS neural network a vector of four values is maintained. This "importance" vector (see Fig. 1) represents the accuracy of each revision subsystem with respect to a class. During the revision, the "importance" vector associated to the class to which the problem case belongs, is used to ponder the outputs of each of the fuzzy revision system. Each value of the vector is associated to one of the four revision subsystems. For each forecasting cycle, the value of the importance vector associated to the most accurate revision subsystem is increased and the other three values are proportionally decreased. This is done to give more relevance to the most accurate revision subsystem.

The revised forecast is then retained temporarily in the forecast database. When the real value of the concentration of pseudo-nitzschia spp is measured, the forecasted value for the variable can then be evaluated, by comparison of the actual and forecasted value, and the error obtained. A new case, corresponding to

CBR-STAGE	Technology	Input	Output	Process
Retrieval	GCS network.	Problem descriptor.	*K* cases.	All the cases that belong to the same class to which the GCS associates the Problem case.
Reuse	RBF network.	Problem descriptor.	Initial solution: concentration of pseudo-nitzschia spp.	The RBF network is retrained with the K retrieved cases.
Revision	4 Fuzzy systems.	Problem descriptor.	Confirmed solution: concentration of pseudo-nitzschia spp.	Four fuzzy sistems are created using the RBF network configuration with different degrees of generalization.
Retain	GCS network. RBF network. 4 Fuzzy systems.	Problem descriptor. Forecasting Error.	Configuration parameters of the GCS network, RBF network and 4 Fuzzy systems.	The configurations of the GCS network, the RBF network and the Fuzzy subsystems are updated according to the accuracy of the forecast.

Fig. 2. Summary of technologies employed by the hybrid model

this forecasting operation, is then stored in the case base. The forecasting error value is also used to update the importance vector associated to the revision subsystems of the retrieved class.

4 Results

The hybrid forecasting system has been proven in the coast of north west of the Iberian Peninsula with data collected by the CCCMM from the year 1992 until the present time. The prototype used in this experiment was set up to forecast the concentration of the pseudo-nitzschia spp diatom of a water mass situated near the coast of Vigo (geographical area A0 ((42°28.90' N, 8°57.80' W) 61 m)), one week in advance. Red tides appear when the concentration of pseudo-nitzschia spp is higher than 100.000 cel/liter. Although the aim of this experiment is to forecast the value of this concentration, the most important objective is to identify in advance if the concentration is going to be over this threshold.

The average error in the forecast was found to be 26,043 cel/liter and only 5.5% of the forecasts had an error higher than 100,000 cel/liter. Although the experiment was carried out using a limited data set, it is believed that these error value results are sufficiently representative to be extrapolated over the whole coast of the Iberian Peninsula.

Two situations of special interest are those corresponding to the *false alarms* and the *undetected blooms*. The first one happens when the model predicts bloom (concentration of pseudo-nitzschia $\geq$ 100,000 cel/liter) and this doesn't take place (real concentration $\leq$ 100,000 cel/liter). The second, more important, arise when bloom really exists and the model doesn't detect it.

Table 2 shows the predictions carried out with success (in absolute value and %) and the erroneous predictions differentiating the undetected blooms and the false alarms. This table also shows the average error obtained with all the techniques. As can be seen, the combination of different techniques in the form of the hybrid CBR system previously presented, produces better results that a RBF neural network working alone or any of the tested statistical techniques. This is due to the effectiveness of the revision subsystem and the retrained of the RBF neural network with the cases recovered by GCS network. The hybrid system is more accurate than any of the other techniques studied during this investigation.

Table 2. Summary of results forecasting pseudo-nitzschia spp

Method	Correct predictions	% Correct	Undetected blooms	False alarms	Average error (cel/liter)
CBR-ANN-FS	**191/200**	**95.5%**	**8**	**1**	**26,044**
RBF	185/200	92.5%	8	7	45,654
ARIMA	174/200	87%	10	16	71,918
Quadratic Trend	184/200	92%	16	0	70,354
Moving Average	181/200	90.5%	10	9	51,969
Simp. Exp. Smooth.	183/200	91.5%	8	9	41,943
Lin. Exp. Smooth.	177/200	88.5%	8	15	49,038

5 Conclusions

This paper has presented a problem solving method in which a CBR system is integrated with two artificial neural networks and a set of fuzzy inference systems in order to create a real-time, autonomous forecasting system. The forecasting system is able to produce a forecast with an acceptable degree of accuracy.

The method uses a CBR system to wrap a growing cell structures network (to index, organize and retrieve relevant data), a radial basis function network (that contributes with generalization, learning and adaptation capabilities) and a set of Sugeno fuzzy models (acting as experts that revise the initial solution) to provide a more effective prediction. The resulting hybrid system thus combines complementary properties of connectionist and symbolic AI methods. The results obtained may be extrapolated to provide forecasts further ahead using the same technique, and it is believed that successful results may be obtained. However, the further ahead the forecast is made, the less accurate the forecast may be expected to be. In conclusion, our hybrid approach to problem solving provides an effective strategy for forecasting in an environment in which the raw data is derived from the previously mentioned sources.

References

1. Tomczak, M., Godfrey, J. S.: Regional Oceanographic: An Introduction. Pergamon, New York, (1994)
2. Fernández, E.: Las Mareas Rojas en las Rías Gallegas. Technical Report, Department of Ecology and Animal Biology. University of Vigo, (1998)
3. Corchado, J. M., Fyfe, C.: Unsupervised Neural Network for Temperature Forecasting. Artificial Intelligence in Engineering, 13, num. 4, (1999) 351–357
4. Corchado, J. M., Lees, B.: A Hybrid Case-based Model for Forecasting. Applied Artificial Intelligence, 15, num. 2, (2001) 105–127
5. Corchado, J. M., Lees, B., Aiken, J.: Hybrid Instance-based System for Predicting Ocean Temperatures. International Journal of Computational Intelligence and Applications, 1, num. 1, (2001) 35–52
6. Corchado, J. M., Aiken, J., Rees, N.: Artificial Intelligence Models for Oceanographic Forecasting. Plymouth Marine Laboratory, U.K., (2001)
7. Nakhaeizadeh, G.: Learning prediction of time series. A theoretical and empirical comparison of CBR with some other approaches. Proceedings of First European Workshop on Case-Based Reasoning, EWCBR-93, Kaiserslautern, Germany, (1993) 65–76
8. Lendaris, G. G., Fraser, A. M.: Visual Fitting and Extrapolation. Weigend, A. S., Fershenfield, N. A. (Eds.). Time Series Prediction, Forecasting the Future and Understanding the Past. Addison Wesley, (1994) 35–46
9. Lekkas, G. P., Arouris, N. M., Viras, L. L.: Case-Based Reasoning in Environmental Monitoring Applications. Artificial Intelligence, 8, (1994) 349–376
10. Faltings, B.: Probabilistic Indexing for Case-Based Prediction. Proceedings of Case-Based Reasoning Research and Development, Second International Conference, ICCBR-97, Providence, Rhode Island, USA, (1997), 611–622
11. Mcintyre, H. S., Achabal, D. D., Miller, C. M.: Applying Case-Based Reasoning to Forecasting Retail Sales. Journal of Retailing, 69, num. 4, (1993), 372–398
12. Stottler, R. H.: Case-Based Reasoning for Cost and Sales Prediction. AI Expert, (1994), 25–33
13. Weber-Lee, R., Barcia, R. M., Khator, S. K.: Case-based reasoning for cash flow forecasting using fuzzy retrieval. Proceedings of the First International Conference on Case-Based Reasoning, ICCBR-95, Sesimbra, Portugal, (1995), 510–519
14. Corchado, J. M., Lees, B., Fyfe, C., Ress, N., Aiken, J.: Neuro-adaptation method for a case based reasoning system. Computing and Information Systems Journal, 5, num. 1, (1998), 15–20
15. Pal, S. K., Dilon, T. S., Yeung, D. S.: Soft Computing in Case Based Reasoning. Springer Verlag, London, (2000)
16. Azuaje, F., Dubitzky, W., Black, N., Adamson, K.: Discovering Relevance Knowledge in Data: A Growing Cell Structures Approach. IEEE Transactions on Systems, Man and Cybernetics, 30, (2000) 448–460
17. Fritzke, B.: Fast learning with incremental RBF Networks. Neural Processing Letters, 1, num. 1, (1994) 2–5
18. Jin, Y., Seelen, W. von., Sendhoff, B.: Extracting Interpretable Fuzzy Rules from RBF Neural Networks. Internal Report IRINI 00-02, Institut für Neuroinformatik, Ruhr-Universität Bochum, Germany, (2000)

Improved Learning for Hidden Markov Models Using Penalized Training

Bill Keller and Rudi Lutz

Computer Science and Artificial Intelligence
The University of Sussex
{billk,rudil}@cogs.susx.ac.uk

Abstract. In this paper we investigate the performance of penalized variants of the forwards-backwards algorithm for training Hidden Markov Models. Maximum likelihood estimation of model parameters can result in over-fitting and poor generalization ability. We discuss the use of priors to compute maximum *a posteriori* estimates and describe a number of experiments in which models are trained under different conditions. Our results show that MAP estimation can alleviate over-fitting and help learn better parameter estimates.

1 Introduction

Hidden Markov Models (HMMs) find wide application to problems of statistical modeling of data sequences, including speech recognition [8,10], language processing [4] and computational biology [6]. A central problem is that of learning a set of model parameters to provide a better fit to the available training data. The algorithm often used is a special case of the expectation maximization (EM) procedure [5] known as the Baum-Welch or forwards–backwards algorithm [1]. This attempts to find a maximum likelihood estimate (MLE) for the model parameters; i.e. a parameter set maximizing the probability of the training data. However, as is well known, the MLE may lead to over-fitting of the training data and poor generalization.

In this paper we investigate the problem of maximum *a posteriori* (MAP) estimation for HMMs using penalized variants of the forwards-backwards algorithm. Rather than selecting the parameters that maximize the probability of the data, on this approach we select the parameters that are most probable given the data. We describe a study in which HMMs are used to model sequences of part of speech (POS) tags drawn from a corpus of natural language text. Language modeling is an important application area for HMMs. To the authors' knowledge, this is the first time that MAP estimation for HMMs has been studied in this context.

2 Hidden Markov Models

We consider only the discrete case. A discrete HMM can be understood as a probabilistic finite state machine, with state set $S = \{s_1, s_2, \ldots, s_N\}$ and output

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 53–60, 2002.

(observation) vocabulary $V = \{v_1, v_2, \ldots, v_M\}$. At each instant of time t, the model occupies a particular state q_t. At time $t = 1$, the model starts out in some state $q_1 = s_i$, with probability π_i. If the model is in state $q_t = s_i$ at time t, then at time $t + 1$ it emits an observation symbol $o_t = v_k$ with probability $b_i(k)$ and switches to a new state $q_{t+1} = s_j$, with probability a_{ij}.

To completely characterize a Hidden Markov Model it is necessary to specify the set of states and observation symbols, and to fix the model parameters $\theta = (\pi, A, B)$. For all i, j, k such that $1 \leq i, j \leq N$ and $1 \leq k \leq M$:

- $\pi = \{\pi_i\}$ where $\pi_i = P[q_1 = s_i]$ – *initial distribution*,
- $A = \{a_{ij}\}$ where $a_{ij} = P[q_{t+1} = S_j | q_t = S_i]$ – *transition probabilities*
- $B = \{b_i(k)\}$ where $b_i(k) = P[O_t = v_k | q_t = S_i]$ – *emission probabilities*

Let $O = o_1 o_2 \ldots o_T$ be a sequence of $T >= 0$ observation symbols and $Q = q_1 q_2 \ldots q_{T+1}$ a sequence of $T + 1$ states. The likelihood $P_\theta(O, Q)$ is the probability of traversing state sequence Q while emitting observations O:

$$P_\theta(O, Q) = \pi_{q_1} \prod_{t=1}^{T} a_{q_t\ q_{t+1}} b_{q_t}(o_t)$$

The probability of observation sequence O is obtained by summing over all possible state sequences: $P_\theta(O) = \sum_{Q \in S^{|O|+1}} P_\theta(O, Q)$.

We may express $P_\theta(O, Q)$ in a different form. Let $c(s_i, s_j; O, Q)$ denote the number of times that a transition from the ith to the jth state occurs in the state sequence Q. Similarly, let $c(s_i, v_k; O, Q)$ denote the number of times that observation v_k is emitted from the ith state. We can then write:

$$P_\theta(O, Q) = \pi_{q_1} \prod_{i,j} a_{ij}^{c(s_i, s_j; O, Q)} \prod_{i,k} b_i(k)^{c(s_i, v_k; O, Q)}$$

The log-likelihood function $L(\theta) = \ln P_\theta(O, Q)$ is expressed as:

$$L(\theta) = \ln \pi_{q_1} + \sum_{i,j} c(s_i, s_j; O, Q) \ln a_{ij} + \sum_{i,k} c(s_i, v_k; O, Q) \ln b_i(k)$$

As will be seen, the EM procedure searches for a set of parameters maximizing the expected value of this function.

3 Parameter Estimation for HMMs

Given a data sequence O, a key problem is to find a parameter set $\hat{\theta}$ such that

$$\hat{\theta} = argmax_\theta P_\theta(O)$$

This is the maximum likelihood estimate (MLE). Intuitively, it yields a model that fits the available data as closely as possible. In general, we do not know which state sequence Q was responsible for generating the observation sequence

O, so it is not possible to solve for $\hat{\theta}$ directly. However, there is an iterative learning technique known as *expectation maximization* (EM) [5], which can be used to find a local, if not always a global optimum in such cases. EM proceeds by maximizing the expected value of the log-likelihood function $L(\theta)$, conditional on the observed data.

More precisely, starting from an initial guess θ^0 at the ML parameters, the following E(xpectation) and M(aximization) steps are repeated until there is no further improvement in the parameter estimates:

E-Step: Calculate $E\{L(\theta)|O, \theta^k\}$ – the conditional expectation of the log likelihood $L(\theta)$ given the observed data O and current parameters θ^k;
M-Step: Set $\theta^{k+1} = \text{argmax}_\theta E\{L(\theta)|O, \theta^k\}$ – the ML estimate for the parameters given the conditional expectation computed in the E-step;

It can be shown that the procedure will eventually converge [5].

In the case of HMMs, the EM procedure is implemented as the forwards-backwards algorithm. The details of this algorithm are not important here. However, the M-step yields the following set of parameter updates:

$$\pi_i = \frac{\sum_Q [q_1 = s_i] P_\theta(Q|O)}{\lambda_\pi}$$

$$a_{ij} = \frac{\sum_Q P_\theta(Q|O) c(s_i, s_j; Q, O)}{\lambda_{A_i}}$$

$$b_i(k) = \frac{\sum_Q P_\theta(Q|O) c(s_i, v_k; Q, O)}{\lambda_{B_i}}$$

The right-hand side numerators correspond to expected counts for various events: e.g. for $b_i(k)$, this is the expected number of times that v_k is emitted by state q_i. The algorithm gets its name from the use of the forwards and backwards variables, which permit these counts to be computed in an efficient way. The denominators are normalizing constants (Lagrange multipliers).

4 Maximum A Posteriori (MAP) Estimation

A problem with ML estimation is that it can lead to over-fitting of the available data and poor generalization performance. To see why, note that the best possible model on this approach would simply memorize the available training data. Such a model would fail to assign any probability mass to data outside the training set. The usual solution to this problem is stopped training: a portion of the available training data is reserved for evaluation, and training is stopped when model performance is observed to degrade on this set. However, it is difficult to justify this procedure on statistical grounds. An appealing alternative is to try to find the set of parameters that is most probable given the training data – the maximum *a posteriori* (MAP) estimate. That is, we would like to find $\hat{\theta}$ such that:

$$\hat{\theta} = argmax_\theta P(\theta|Q, O)$$

For $P(\theta)$ a prior, then from Bayes law, $P(\theta|Q,O) = P_\theta(Q,O)P(\theta)/P(Q,O)$. Since $P(Q,O)$ is constant, we may as well find the maximum of the right hand numerator. Equivalently, we should maximize the sum:

$$\ln P_\theta(Q,O) + \ln P(\theta)$$

This is just the log-likelihood function plus a penalty term (the log of the prior distribution). The penalized EM (PEM) procedure is applied as follows:

E-step: Calculate $E\{L(\theta)|O,\theta^k)\}$ – as for the standard EM procedure.
M-step: Set $\theta^{k+1} = \text{argmax}_\theta E\{L(\theta)|O,\theta^k\} + \ln P(\theta)$ – the MAP estimate given the conditional expectation computed in the E step.

Thus, the only change is in the M-step, though in practice it may be difficult to find a prior that allows us easily to carry out the maximization.

5 Selecting a Prior Distribution

Our interest is in selecting priors that aid the learning process e.g. by helping to avoid overfitting. One possibility is to consider priors that encode given information about the learning task. This might be information implicit in the training data, or knowledge provided by a domain expert. Another possibility is to adopt a form of 'Occam's Razor', and select priors that favour simpler or smaller models. The priors considered in the present study are described below: a Dirichlet prior utilizing symbol frequency information, and two priors with corresponding 'entropic' penalty terms.

5.1 Dirichlet Distributions and the "Symbol Frequency" Prior

For a discrete probability distribution $\mathbf{p} = p_1, p_2, \ldots, p_m$ the Dirichlet distribution has the form:

$$P(\mathbf{p}) = \frac{1}{C}\prod_{i=1}^{m} p_i^{\alpha_i - 1}$$

where the α_i are the distribution parameters and C is a normalizing constant. Greater probability mass is assigned to distributions $\mathbf{p}$ that reflect the relative magnitudes of the $\alpha_i - 1$ terms. The 'weight' of the distribution is $\sum_i \alpha_i$: the larger the weight, the stronger the bias.

The Dirichlet distribution has previously been proposed in bioinformatics to encode prior knowledge of the distribution of amino acids in models of gene homology [3]. Stolcke and Omohundro [11] have used Dirichlet priors as part of their state-merging technique for inducing HMMs. We have used the Dirichlet distribution to fix a prior over the emission probabilities. The parameter update rule becomes:

$$b_i(k) = \frac{\sum_Q P_\theta(Q|O)\, c(s_i, v_k; Q, O) + c_k}{\lambda_{B_i}}$$

where the new term c_k in the numerator may be regarded as an extra 'count'. In the experiments reported in the next section the c_k were set to the normalized frequencies of the observations v_k in the training data. In effect, this provides a bias only in the absence of information from the data (i.e. where the expected counts returned by the forwards–backwards algorithm are very low).

5.2 Entropic Priors

Brand [2] reports work on HMM training using priors of the general form:

$$P(\mathbf{p}) = \frac{1}{C} \prod_{i=1}^{m} p_i^{\kappa p_i}$$

where $\kappa = \pm 1$. The associated penalty term is:

$$\ln P(\mathbf{p}) = -\ln C + \kappa \sum_{i=1}^{m} p_i \ln p_i$$

which is entropic in form. Maximizing $\ln P(\mathbf{p})$ will maximize ($\kappa = -1$) or minimize ($\kappa = 1$) the entropy of the distribution $\mathbf{p}$. We might expect maximizing entropy to have a 'smoothing' effect on parameter distributions that would help avoid over-fitting. On the other hand, Brand [2] reports good results for minimization of entropy. In particular, this appears to reduce model complexity by helping to 'zero' parameters.

Entropic priors were applied to both the state transition and symbol emissions probability distributions. A problem that arises in this case is that the modified M-step cannot be carried out exactly. However, it is possible to derive a set of fixed-point equations, as shown below for the transition probabilities A:

$$a_{ij} = \frac{c(s_i, s_j; Q, O) + \kappa a_{ij} \ln a_{ij}}{\sum_{k=1}^{N} (c(s_i, s_k; Q, O) + \kappa a_{ik} \ln a_{ik})}$$

In the implementation of the M-step, these equations are used iteratively to compute a solution of the desired accuracy.

6 Experimental Study

Our interest in HMMs lies in their application to statistical language modeling. We used part-of-speech (POS) tagged natural language text as our training data. However, it is important to note that the results reported in Sect. 7 appear to be quite general, and should hold for sequence data drawn from other application domains.

6.1 The Data

As experimental data we used the first 5000 POS tagged sentences of the written section of the British National Corpus (BNC). All lexical tokens were removed to produce simple tag sequences. The 5000 sentences were split into two parts: 2500 sentences to be used as training data, and the remaining 2500 as test data. To permit 10-fold cross-validation, the training sequences were further split into 10 pairs of training/evaluation sets. Each pair comprised a unique evaluation set of 250 sequences, and a corresponding training set formed from the other 2250 sequences.

The entire data set contains 84 POS tags. Each of the 10 training/evaluation sets contains all of these 84 symbols, as does the test set. (This avoids any problem of symbols in the test data not present in the training data). The smaller evaluation data sets do not individually cover the entire symbol set, ranging from just 76 symbols to 83.

6.2 The Experiments

Experiments were conducted with fully connected HMMs having 10, 15, 25, 50 and 100 states. At each model size, we carried out 10 sets of training runs using different priors: **NoPenalty** (no prior – the standard case), **SF** (the Dirichlet "symbol frequency" prior on emissions), **MaxEtMaxEe** (maximum entropy on both emissions and transitions) **MinEtMinEe** (minimum entropy on both emissions and transitions). In addition, we experimented with combinations of these, e.g. **MaxEtSF**: maximum entropy on just transitions combined with the symbol frequency prior.

One set of runs was conducted for each of the 10 pairs of training and evaluation data. To permit fair comparison later, for a given pair each run was started from the same (randomly initialized) HMM. For model sizes of 10 to 50 states the training runs each consisted of 600 iterations. The time taken to complete training for the 100 state models meant that runs of only 200 iterations were produced. For each run and at each iteration we recorded the cross-entropy of the model on the training data, and also on the evaluation data. We also recorded the 'best' model from each run (i.e. the model with lowest perplexity on the evaluation data) and the final model produced.

7 Results

The experimental data produced by the study showed little variation across model sizes. Figure 1 graphs average performance for a number of priors at the 50 state model size, and is quite typical of the results produced at other model sizes (modulo some change of scale). Averages are shown only for **NoPenalty**, **SF**, **MaxEtMaxEe** and **MaxEtSF**. Performance of **MinEtMinEe** was generally worse than **NoPenalty** and has been excluded. The lower curves show average cross-entropy on the training set as a run proceeds and are typical for EM, but it

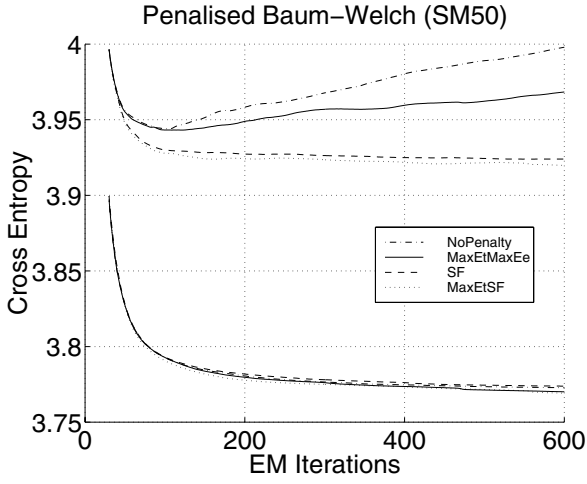


Fig. 1. Average performance at model size 50

is notable that there is apparently little variation between the averages of different training regimes. On the evaluation data there appear to be clear differences however, with both **NoPenalty** and **MaxEtMaxEe** over-fitting, while **SF** and **MaxEtSF** show continuous improvement over the course of the run. Further, it appears that both **SF** and **MaxEtSF** result in better models overall.

The average performance illustrated in Fig. 1 masks variation across individual runs. However, much of this variation may be accounted for by differences in the training conditions (i.e. the particular training and evaluation sets, and the initial HMM used for each run). To test statistical significance, we applied a one-tailed t-test for paired observations. At the 95% confidence level ($\alpha = 0.05$) this showed that **SF** and **MaxEtSF** perform significantly better than both **NoPenalty** and **MaxEtMaxEe** at all points after about 50 iterations. On the other hand, no consistent improvement was detected between either **NoPenalty** and **MaxEtMaxEe**, or **SF** and **MaxEtSF**.

Finally we examined the relative performance of the best models saved from each run. At a given model size, for each prior we took the best model saved on each of the 10 runs (R1 to R10) and measured its actual performance on the 2500 sequence test set. The results for the 50 state model size are summarized in Table 1 where performance if given in terms of perplexity on the test set. The best model in each column is indicated in bold face.

8 Conclusions

We have investigated the use of penalized variants of the forwards-backwards algorithm for discrete HMM training. The results appear to be robust across model sizes and variations in initial conditions. Using prior information about

Table 1. Best model performance at the 50 state model size

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
NoPenalty	14.45	14.42	14.37	14.25	14.16	14.17	13.97	14.27	14.76	14.68
MaxEtMaxEe	14.26	14.59	14.37	14.26	14.14	14.42	13.93	14.22	14.64	14.64
SF	14.04	14.16	14.21	**14.00**	**13.85**	13.90	13.64	**14.02**	**14.22**	14.28
MaxEtSF	**14.02**	**13.95**	**14.18**	**14.00**	13.88	**13.84**	**13.62**	14.05	14.24	**14.23**

the relative frequencies of symbols in the training data has been shown to have a beneficial effect. Both the **SF** and **MaxEtSF** priors help avoid over-fitting and lead to better solutions overall. Although maximizing entropy may help somewhat by 'smoothing' the A and B parameter distributions, the benefits do not appear to be great. We found that minimizing entropy can exacerbate the problem of over-fitting. This is perhaps surprising in view of the results reported in [2] and deserves further investigation. Currently, we are looking in more detail at the use of Dirichlet priors. In particular, we are investigating the use of "symbol frequency" priors of different weights, and the use of "flat" Dirichlet priors over the state transition probabilities.

References

1. Baum, L.E., and Petrie, T. (1966). Statistical inference for probabilistic functions of finite Markov chains. *Annals of Mathematical Statistics*, **41**, 164–171.
2. Brand, M. (1999). Structure learning in conditional probability models via an entropic prior and parameter extinction. *Neural Computation*, **11**(5), 1155–1182.
3. Brown, M., Hughey, R., Krogh, A., Mian, I.S., Sjölander, K., and Haussler, D. (1993). Using Dirichlet mixture priors to derive Hidden Markov Models for protein families. *Proceedings First International Conference on Intelligence Systems for Molecular Biology*, 47–55.
4. Cutting, D., Kupiec, J., Pedersen, J., and Sibun, P. (1992) A practical part-of-speech tagger. *Proceedings, Third Conference Applied Language Processing*, Trento, Italy, 133-140.
5. Dempster, A.P., Laird, N.M., and Rubin D.B. (1977). Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society B*, **39**, 1–38.
6. Eddy, S.R. (1998) Profile Hidden Markov Models. *Bioinformatics*, **14**(9), 755–763.
7. Green, P.J. (1990b) On the use of the EM algorithm for penalized likelihood estimation. *Journal of the Royal Statistical Society B*, **52**, 443–452.
8. Jelinek, F. (1976) Continuous speech recognition by statistical methods. *IEEE*, **64**(4), 532–556.
9. McLachlan, G.J. and Krishnam T. (1997). The EM Algorithm and Extensions. John Wiley and Sons, Inc, New York.
10. Rabiner, L.R. (1989) A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, **77**(2), 257–285.
11. Stolcke, A. and Omohundro, S. (1994). Best-first model merging for hidden Markov model induction. Technical Report TR-94-003, International Computer Science Institute, 1947 Center St., Berkley, CA, 94704, USA.

Recovering High-Level Structure of Software Systems Using a Minimum Description Length Principle

Rudi Lutz

School of Cognitive and Computing Sciences
University of Sussex
rudil@cogs.susx.ac.uk

Abstract. In [12] a system was described for finding good hierarchical decompositions of complex systems represented as collections of nodes and links, using a genetic algorithm, with an information theoretic fitness function (representing complexity) derived from a minimum description length principle. This paper describes the application of this approach to the problem of reverse engineering the high-level structure of software systems.

1 Introduction

Reverse engineering the high-level structure of a piece of code is an important task in software engineering. There are large numbers of so-called "legacy" systems in use, where the documentation is either out-of-date, or non-existent. Under these circumstances, where it is required to either maintain or improve the system (but there is insufficient documentation) an important first step is to try to recover the high-level structure of the code.

A common approach to this task is to start with a graph representing various dependency relationships that hold in a program. This can be a graph of the calling relationships between procedures, augmented by information such as variable usage [8], or a graph of dependencies between existing "modules" (usually files) [13,3]. One can then try to recover higher level system structure by attempting to find groupings of nodes in the dependency graph which "naturally" go together to form functionally cohesive units. There have been two main approaches to this grouping task – one based on defining some sort of similarity metric for nodes in the graph (typically based on similarities in their interconnection pattern), and then to cluster the nodes in the graph using a (possibly hierarchical) clustering algorithm (see [20] for a nice overview). The other approach [13, 3,6] (of which the work reported here is an example) involves defining a "goodness" criterion for (possibly hierarchical) decompositions of the graph, and then to search the space of possible decompositions using some sort of search algorithm, such as simulated annealing [9], genetic algorithms [15], or some more classical search algorithm [19].

However, the search technique is in some sense of secondary interest compared to the answer given to the really fundamental question: *How do we tell if one decomposition into modules is better than another?* Most of the work in this field (e.g. [6,13,3]) has taken the view that one decomposition is better than another to the extent that it does a better job of minimizing the coupling between modules, and maximizing the cohesion

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 61–69, 2002.

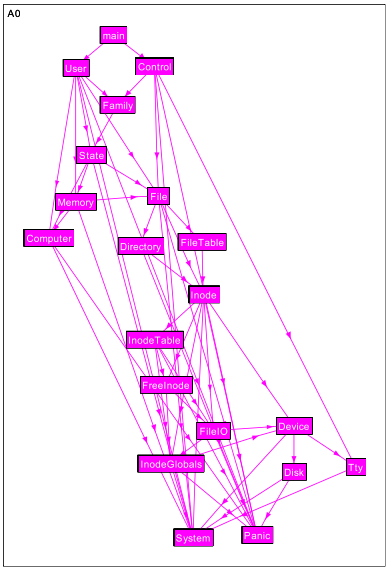


Fig. 1. A software system (nodes and links)

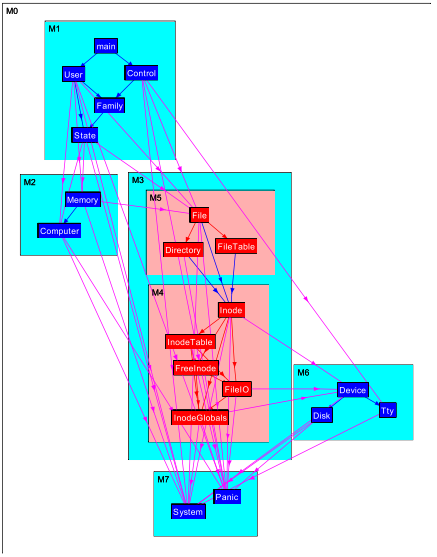


Fig. 2. A modular software system

within modules, where coupling is the extent to which dependencies between software components cross module boundaries, and cohesion is the extent to which components inside a given module are related to each other.

However we take a slightly different view, and regard the aims of reducing coupling and increasing cohesion as in some sense secondary. What we are really aiming for is the decomposition that makes the system as simple as possible. In this paper, based on work described in [12,21], we use as our fitness function, a complexity measure based on a *minimum description length* (MDL) principle, and loosely drawing on ideas from Kolmogorov Complexity Theory [11]. This amounts to choosing that hierarchical modular decomposition (HMD) which is the easiest to describe i.e. has the shortest description. The MDL Principle has been used for some time in machine learning work (see for example [16]), but this is the first use of it as a guiding principle for reverse engineering. This paper therefore aims to give a brief account of our approach, to present the results of one of our first experiments using it, and to compare these with those of the BUNCH system [13,3], which is very similar in spirit to this work, but uses measures of coupling and cohesion in order to perform the clustering.

2 Systems, Graphs, and HMDs

Motivated by the work of Briand et al. [1] we take a system to be a directed graph consisting of a collection of nodes and links between them. An example is shown in Fig. 1. In such a graph the nodes represent program entities, and the links represent dependency relationships between these entities. A possible decomposition of this system is shown in Fig. 2. Given a system S we define a *hierarchical modular decomposition*(HMD) of

S to be a pair $\langle T, E \rangle$ consisting of the edges E of S together with a *module tree* T. A module tree for a system S is a tree T whose leaves are the nodes of S. Any non-leaf nodes of T correspond to modules of S, and the daughter relationship between two nodes denotes the obvious module/sub-module or module/node relationship. A module tree corresponding to Fig. 2 is shown in Fig. 3.

3 The MDL Principle

Kolmogorov Complexity Theory [11] defines the complexity of a (computable) object as the information content of the object due to its structure alone. The information content of an object is in turn defined as the length *in bits* of the shortest Universal Turing Machine program that can compute the object. Such a program will consist of the description of a Turing machine plus its data. This measure of complexity can then be shown to be independent (up to an additive constant) of the choice of Universal Turing machine. Unfortunately Kolmogorov Complexity is uncomputable in general. So, in practice one usually tries to devise an encoding scheme for the objects that one expects to be reasonably minimal, and uses the length (in bits) of this encoding as the (relative) complexity of the object. Fixing an encoding scheme amounts to fixing the Turing machine, and the encoding of the object is then its data.

Now, given some data D, and a possible explanation (or description, or theory) H for that data, an interesting question is: *What is the best (simplest) explanation or description of the data?* The *Minimum Description Length (MDL) Principle* [17] answers this by asserting that the best description is the one which minimises the total complexity of the theory and data i.e. the combined description length of the explanation H and that of the data (now re-described using H). In our case the H will be a possible HMD of some system (D), and since H completely encodes D (D consists of the edges in H, and the leaves of the module tree in H), so that it is unneccessary to redescribe D. We will therefore be looking for the least complex (i.e. shortest description) modularisation.

4 An MDL-Based Complexity Measure for HMDs

The basic idea is that the complexity of a HMD is given by the length of a *decodeable* bitstring describing the HMD. Note that we do not actually need to construct the bitstring, so long as we have a formula for how long it will be. Thinking of the bitstring as a message that is being sent to some recipient who must decode it, enables us to use information theoretic results in order to compute its length. The following discussion will briefly describe the message format so that we can give a formula for the length.

In order to describe an HMD $\langle T, E \rangle$, we first send a description of T, followed by a description of E. So we first send a recursively structured message describing T. One begins by sending a description of the top-level module. Now, for any module m (including the top-level one) one must first state which nodes (leaves of T) are in the module, followed by details of each of its sub-modules. To describe the nodes in a module we first send an integer to say how many nodes there are, and then for each node send a codename that will be used to represent it, so that any reference can be recognised in the link descriptions coming later. To describe the sub-modules we first state how

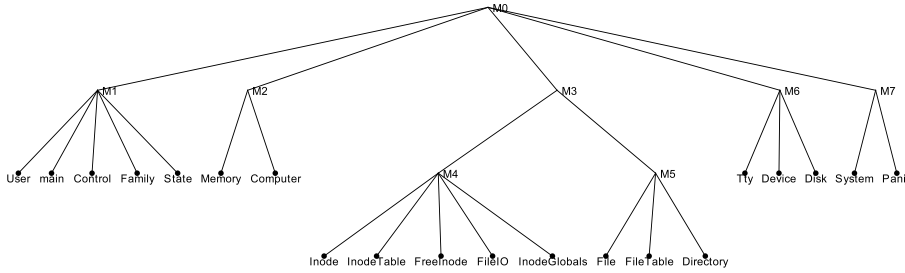


Fig. 3. A module tree for Fig. 2

many sub-modules there are, followed by a codename for, and a (recursively structured) description of, each of its sub-modules.

Having sent details of the module structure we then send a description of all the edges. For each node in the system (in the order in which their codenames appeared in the preceding module description) we send details of the edges emanating from this node. To do this we send an integer saying how many edges this node is the source for, followed by the *relative pathname* for each of the destination nodes that this node is connected to. In the case where the destination node is not in the module, or in a sub-module of the module, in which the source node occurs, then the relative path consists of a symbol saying we have to go up out of this module, and integer saying how far up the tree to go, followed by a sequence of module codenames, and then the codename of the destination node. Alternatively the relative pathname is either just the name of a node (source and destination are in the same module) or a sequence of sub-module names followed by the destination node name.

The codename of a connected node n will occur in the whole message once for each incoming edge of n, and once to specify the code name prior to the edge descriptions, giving a total of $f(n) = indegree(n) + 1$. Similarly, the codename of each module m will occur $f(m) = indegree(m) + 1$ times.

Therefore, a description of a HMD will consist of a sequence of code words, some representing (names of) modules or basic nodes, and some representing integers. Since the codes must be uniquely decodeable, and there is no upper bound on the size of integers that may need to be sent, we must use a prefix code (see [11] for details). One such code enables an integer n to be sent using a code length of:

$$l(n) = \log_2(n) + 2\log_2\log_2(n+1) + 1$$

Now, what is the length of codenames for nodes and modules? It should be noted that, because of the use of relative pathnames above, codenames for nodes and modules need only be unique within their enclosing module. Also, information theory [18] tells us that, when sending a message containing symbols s_i occurring with frequencies f_i, then in order to minimise the message length we can (and must!) use codes of length $-log_2(\frac{f_i}{\sum_i f_i})$ bits, for each s_i. Careful consideration of the number of times each symbol occurs, and what integers need to be sent then leads to the following formula for the

complexity $\psi(X)$ of a HMD X:

$$\sum_{m \in \mathcal{M}} \left(\begin{array}{l} l(|N_m|+1) + l(|M_m|+1) + \sum_{n \in N_m \cup M_m \cup \{up\}} \left[l\left(-\log\left(\frac{f_n}{F_m}\right)\right) - f_n \log\left(\frac{f_n}{F_m}\right) \right] \\ + \sum_{n \in N_m} \sum_{n' \in out\,N(n)} l(relD(n, lca(n, n'))) \end{array} \right)$$

In this formula $\mathcal{M}$ is the set of all the modules in the HMD X, and $F_m = \sum_{n \in N_m \cup M_m \cup \{up\}} f_n$. We note that the first occurrence of each of the codenames has to be preceded by its length, so the recipient can tell when the name ends. This gives rise to the $l(-\log\left(\frac{f_n}{F_m}\right))$ terms. The $l(relD(n, lca(n, n')))$ term corresponds to the integers saying how far up the tree to go – $relD(n, m)$ gives the relative depth between a node n and an enclosing module m, and $lca(n, n')$ is the *least common ancestor* (deepest enclosing module) of the nodes n and n' (see [12] for further details). Note that we follow the usual practice of allowing "fractional bits"(which give the theoretical lower bounds on the number of bits needed) as this seems to give the best results.

Although at first sight it would seem that the minimum length message would correspond to a "flat" HMD with just a single module (as shown in Fig. 1), this is not the case. Although such an HMD may have the smallest number of high-level symbols (node and module names, and integers) in its description, it does not necessarily have the shortest length *in bits*. This is because the codes for the symbols are chosen locally within a module , so one is essentially using a smaller alphabet of symbols, and the codes are shorter as a result. Additionally, more frequently used names get shorter codelengths. So there is a trade-off between the costs of describing modules and resulting module boundary crossing, and the ability to use shorter codes for each symbol. This is what makes this approach both interesting, and subtle in its effects.

5 The Genetic Algorithm

The number of ways of arranging a system as a set of nested modules is enormous, and finding the optimal HMD for a system thus involves a search through a very large space (with many local optima). In [21] this was searched using a heuristically based beam search [19]. This proved to be rather slow, and often had trouble finding the best HMD, since in essence it was using a local-search based hill-climbing strategy. A Genetic Algorithms (GAs) [5,15,7] are a family of search techniques that are often very effective in such large difficult search spaces. We therefore have used a GA for this work.

In this work we have used a distributed "steady-state" GA [2] (rather than the more traditional"classical" generational GA). In this type of GA the population is spatially distributed (usually over a 2-dimensional grid), and individuals only "mate" with those in some neighbourhood of themselves. Additionally, children are immediately put back into the population when they are created, rather than waiting until an entire new population of children has been produced. In many domains (see for example [14]) these outperform the classical GA, perhaps because the distributed nature of the population allows different solutions to arise in spatially separated areas of the grid, helping to prevent premature convergence of the population. It would be an interesting future experiment to compare a variety of different styles of GA on our problem.

Reasons of space will not permit a full discussion of the mutation and crossover operators used in this work. Readers interested in full details are referred to [12]. However they have much in common with similar operators in the Genetic Programming (GP) field [10]. In particular, there are several mutation operators, each applied with some specific probability, and which do such things as move a node (either a module, or leaf node) from one place in a module tree to another, introduce an extra module node, or delete a module node, making the children of the deleted node into children of the parent of the deleted node. Similarly, crossover involves choosing a random subtree in each of the two module trees involved and interchanging them. However, this does not necessarily give us a legal module tree (some leaf nodes might not occur at all, others might occur twice) and it may have to be "repaired" (see [12] for details).

Let S be some system for which we are trying to find the best HMD. The genome for an individual X in the GA simply consists of a HMD for S. The fitness of X is given by $f(X) = \frac{1}{\psi(X)}$ where $\psi(X)$ is the complexity of the HMD represented by X. Maximizing f will therefore minimize complexity.

6 Reverse Engineering the MiniTunis System

The system we used to test the effectiveness of our approach was the Mini-Tunis system described in [3]. Figure 1 shows the "module dependency graph" for this system, and the system documentation, according to [3] describes the high-level structure of the system as that shown in Fig. 2. We chose this system because it is reportedly a well-structured system, and accordingly the extent to which we can recover the modular structure would be a good test of our approach, and also enable us to compare our approach to that of the BUNCH [13,3] system. The best modularisation found by BUNCH is shown in Fig. 4.

We performed 10 runs of our system on the graph shown in Fig. 1. We used a 20×20 grid, and each run was for 1000 generations (chosen empirically because the "best fitness" stabilised before this). The best modularisation found is shown in Fig. 5 (complexity 530.55 bits). It found this solution in 6 of the 10 runs. The remaining 4 solutions were very similar to this, and it is possible that had the runs continues a little longer the same solution would have been found (the standard deviation $\sigma = 0.623$, indicating the great similarity in the complexity values). At first sight our system seems to have produced a very different modularisation to the original since it has produced a very nested structure. However, if we just look at the *nodes* (and ignore the nesting) in each module we see that that our system has done an extremely good job of grouping nodes together in much the same way as the original. There are two main differences:

- Node *state* has move into the module with *computer* and *memory*. Given only the links in the graph as information this is probably unavoidable (note that BUNCH did this too).
- The system globals *Panic* and *System* have been moved into the same module as *Device* etc. This is because our system has no notion of global immoveable nodes. If we do regard these nodes as global and immoveable, and adopt the point of view that therefore links to these nodes should be "constant cost", then we get the same modularisation as Fig. 5, but with *System* and *Panic* in a top-level module of their own. However this does require prior information that these two nodes are global.

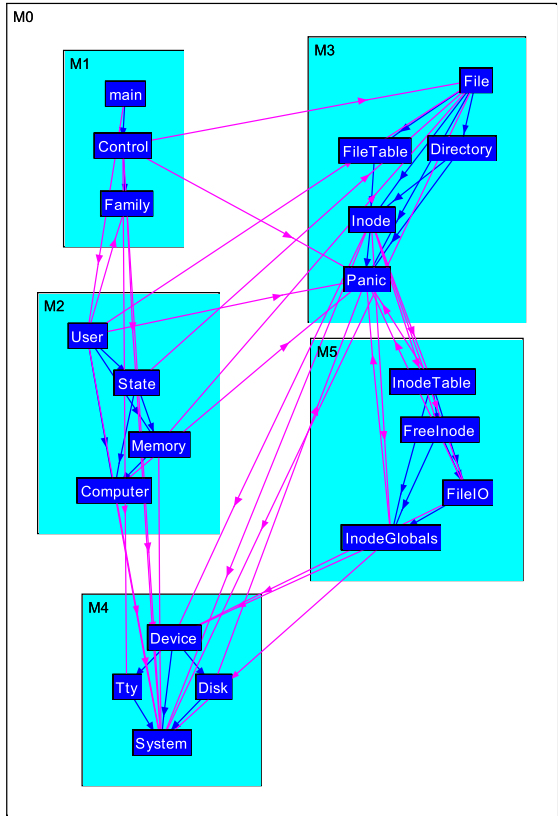


Fig. 4. Best Modularisation Achieved by BUNCH

Apart from these two differences our system clearly has identified groupings of nodes that are almost identical to the original documentation, and has done better *in this respect* than BUNCH. It is also interesting to note that the BUNCH modularisation has a lower complexity according to our metric than does the original, so its attempt to reduce coupling and maximise cohesion has also reduced complexity in our sense.

7 Conclusions

Although this work is still at an early stage it is clear that our complexity metric functions well for the task of reverse engineering system structure, at least for the purposes of deciding which program entities "go together" to form functional units. However, there is perhaps too great a bias towards nesting (even though some input graphs do give rise to unnested structure), which future work will try and address (although it is not completely clear at this time that this is necessarily incorrect!). It also seems that a GA, with our domain specific GP inpired mutation and crossover operators is an interesting technique

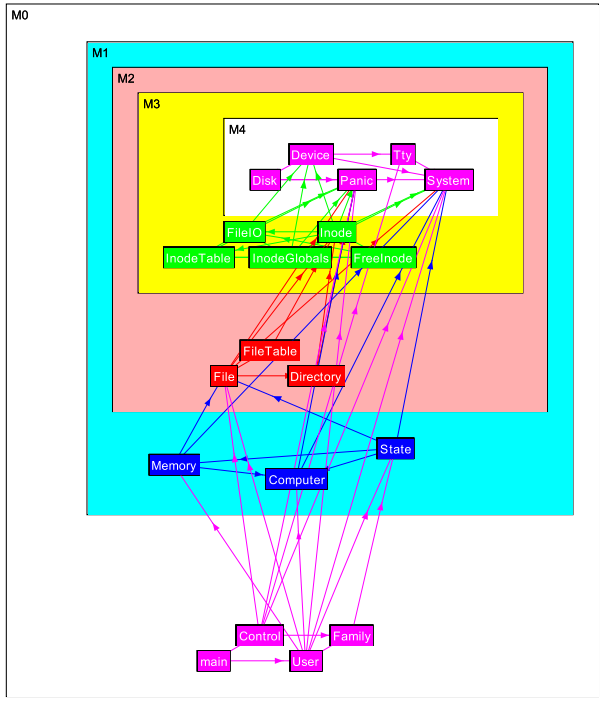


Fig. 5. Evolved Mini-Tunis System

for reverse engineering complex dependency graphs into a hierarchy of functionally meaningful units.

References

1. Briand, L.C., Morasca, S., and Basili, V.R. (1996) Property-based software engineering measurement: Refining the additivity properties. *IEEE Transactions on Software Engineering*, 22(1):68–86.
2. Collins, R. and Jefferson, D. (1991) Selection in massively parallel genetic algorithms. *Proceedings of the Fourth International Conference on Genetic Algorithms, ICGA-91* Belew, R.K. and Booker, L.B. (eds.), Morgan Kaufmann.
3. Doval, D., Mancoridis, S., and Mitchell, B.S. (1999) Automatic Clustering of Software Systems using a Genetic Algorithm. *IEEE Proceedings of the 1999 International Conference on Software Tools and Engineering Practice (STEP'99)*.
4. Glover, F. (1989) Tabu Search - Part I. ORSA Journal on Computing, Vol. 1, No. 3, pp. 190-206.
5. Goldberg, D.E. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.
6. Harman, M., Hierons, R., and Proctor, M. (2002) A New Representation and Crossover Operator for Search-Based Optimization of Software Modularization. Submitted to GECCO–2002.

7. Holland, J.H. (1975) *Adaptation in Natural and Artificial Systems*. Now published by MIT Press.
8. Hutchens, D., and Basili, R. (1985) System Structure Analysis: Clustering with Data Bindings. *IEEE Transactions on Software Engineering*, SE-11(8):749-757, 1985.
9. Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P. (1983) Optimization by Simulated Annealing, Science, 220, 4598, 671-680.
10. Koza, J.R. (1992) *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press.
11. Li, M. and Vitanyi, P. (1997) *An Introduction to Kolmogorov Complexity Theory and Its Applications*. Springer-Verlag.
12. Lutz, R. (2001) Evolving Good Hierarchical Decompositions of Complex Systems. *Journal of Systems Architecture*, **47**, pp. 613–634.
13. Mancoridis, S., Mitchell, B.S., Rorres, C., Chen, Y.,Gansner, E.R. (1998) Using automatic clustering to produce high-level system organizations of source code. In *International Workshop on Program Comprehension (IWPC'98)* IEEE Computer Society Press, Los Alamitos, California, USA, pp.45–53.
14. McIlhagga, M., Husbands, P., and Ives, R. (1996) A comparison of simulated annealing, dispatching rules and a coevolutionary distributed genetic algorithm as optimization techniques for various integrated manufacturing planning problems. In *Proceedings of PPSN IV*, Volume I. LNCS 1141, pp. 604–613, Springer-Verlag.
15. Mitchell, M. (1996) *An Introduction to Genetic Algorithms*. MIT Press.
16. Mitchell, T.M. (1997) *Machine Learning*. McGraw-Hill.
17. Rissanen, J. (1978) Modelling by the shortest data description. *Automatica-J.IFAC*, 14, pp.465–471.
18. Shannon, C.E. (1948) The mathematical theory of communications. *Bell System Technical Journal* 27:379–423, 623–656.
19. Thornton, C.J. and du Boulay, B. (1992) *Artificial Intelligence Through Search*. Intellect, Oxford, England.
20. Wiggerts, T. (1997) Using clutering algorithms in legacy systems remodularisation. In *Proc. Working Conference on Reverse Engineering* (WCRE'97)
21. Wood, J.A. (1998) Improving Software Designs via the Minimum Description Length Principle. Ph.D. Thesis, University of Sussex (available from http://cogslib.cogs.susx.ac.uk)

A System for Multi-agent Information Retrieval

Paula Mc Dermott and Colm O'Riordan

Department of IT, NUI, Galway
{Paula.Mc_Dermott,colmor}@geminga.nuigalway.ie

Abstract. In this paper we discuss a multi-agent information trading system, which we argue provides a suitable and flexible design for distributed information retrieval, and a suitable test-bed for studying agent cooperation, coordination and negotiation. Within the fields of Information Retrieval (IR) and Information Filtering (IF), agents represent a suitable paradigm to conceptualise, design and implement an IR system.

1 Introduction

The purpose of the system described and developed in this paper is twofold. Firstly, the system acts as an agent based information trading system and secondly it forms a framework for testing various cooperation, coordination and coalition formation strategies. Within this framework, agents may have the potential to learn about information needs and other agents abilities etc. As sources become more distributed, we have a set of information providers, with their set of repositories and an associated cost to access this information.

The paper is laid out as follows. Section 2 outlines research in IR and distributed IR. Research and development in MAS is outlined in Sect. 3. The following section discusses the design of our system. Section 5 describes current and future experiments. Section 6 offers results and the paper is concluded in Sect. 7.

2 Information Retrieval

The task of an Information Retrieval system is to retrieve documents containing information that is relevant to a user's information need [1]. This task is a difficult one as it usually deals with natural language which is not always well structured and may be semantically ambiguous [2].

2.1 Approaches

The task of IR can be broken down into three subtasks, namely, representation, comparison and feedback.

Documents and queries must be represented in the same machine readable manner, such that the system can perform comparison techniques between them.

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 70–77, 2002.

Representation techniques range from using indexes, vector and matrix representation to more modern representations, such as neural networks, connectionist networks and semantic networks [3].

The comparison mechanisms used are dependent on the underlying representations employed in the system. There are three classic representations (models) in IR, namely, Boolean [2], Vector [6] and Probabilistic [19].

Due to the nature of IR the initial results will be a partial match to the query. The returned set is improved by query reformulation, which may be performed by the user but is largely automated by IR systems. It involves two steps: expanding the original query with new terms and re-weighting the terms in the expanded query.

2.2 Distributed Information Retrieval

Given the increase in size of information repositories, the need for alternative architectures and algorithms has emerged. Modern information environments have become large, open and heterogeneous [7]. This has led to the increased adoption of both distributed and parallel architectures.

Commonly identified problems within distributed IR are those of source selection (which repository is most appropriate for a given query) and result merging (how to combine results from different sites). Possible solutions for source selection include central indexing mechanisms and rule-based techniques. Voorhees et al. [18] and Callan et al. [17] provide techniques for dealing with result merging.

3 Multi-agent Systems

Software agents date back to the early days of AI work and Carl Hewitt's concurrent actor model [5]. In this model, Hewitt proposed the concept of a self-contained interactive and concurrently executing object which he termed an actor. This object had some encapsulated internal state and could respond to messages from similar objects [4]. Wooldridge [16] provides the following definition: *"An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives".*

Generally a multi-agent system is composed of a number of agents that are able to interact with each other and the environment and that differ from each other in their skills and their knowledge about the environment [8]. The agents in a multi-agent system will need to communicate with each other, be aware of each other and reason about each other. The complexity introduced by agent-agent interactions has created, among others, the research areas of agent cooperation, coordination and coalition formation.

3.1 Agent Traits

Cooperation. Cooperation is often presented as one of the key concepts which differentiates multi-agent systems from other related disciplines such as dis-

tributed computing, object-oriented systems and expert systems [9]. Notions of shared or joint goals and notions of complimentary individual action have been posited. Theories from formal logics [10], philosophy [15] and game theory [14] have been adopted to reason about cooperation and to design and develop models whereby agents cooperate.

Coordination. Coordination is a central issue in software agent systems and in Distributed Artificial Intelligence (DAI) [11]. Coordination is required in a multi-agent system for many reasons, including: preventing anarchy or chaos, meeting global constraints, efficiency, distributed expertise and dependencies between agents' actions. It is important to note, that coordination may require cooperation but cooperation will not necessarily result in coordination. Many coordination strategies have been developed, they can be categorised as follows: Organisational Structuring, Contracting, Multi-Agent Planning and Negotiation [11].

Coalition Formation. A coalition can be defined as a group of agents who have decided to cooperate in order to perform a common task [12]. Shehory, Sycara and Jha [12] believe that the incorporation of a coalition formation mechanism will increase the efficiency of group-wise task execution, resulting in near-optimal task performance. In general agents will only form a coalition if each member of the coalition gains more by joining the coalition than by working alone. Work in this domain ranges from sub-additive coalitions [13] to greedy strategy coalitions [12] and those that are based on agent negotiation.

In recent years, agents have been developed and 'released' into information management environments in an attempt to improve the mechanisms for finding, fusing, using, presenting, managing and updating information.

4 System Design

The primary concern of this paper is the application of agent-based systems to the domain of information management. Weiss [8] describes information agents as: agents that have access to multiple, potentially heterogeneous and geographically distributed information sources. Information agents have to cope with the increasing complexity of modern information environments and retrieve, analyse, manipulate and integrate information from different sources.

4.1 Agent Community

The agents in the system have three main components in their architecture: *knowledge, beliefs and capabilities.*

The agent's knowledge base comprises *user(s)*, *information need(s)* and a *utility* value. The information need represents the task that the user is requesting the agent to carry out. The utility or self-belief value is a measure of the user's

satisfaction with the agent's performance. Depending on the agent's performance the user will increase, or decrease, its trust in the agent through user feedback.

The agent's beliefs represent the agent's opinions of one another. The *belief* rating refers to one agent's level of belief in another agents ability to perform a task. The *trust* rating, on the other hand, relates to an agent's belief that the agent will actually carry out the task. This section is an important part of the agent architecture. The belief and trust levels that an agent has in another agent will be taken into account when an agent is considering who is most suitable for a particular task.

The capabilities section refers to the tasks that the agents will be able to perform. All agents will be capable of carrying out the same tasks (including *retrieval,* and *learning*) but they will differ in their ability to perform them, for example some agents will perform Boolean retrievals while others will perform vector based retrievals.

4.2 Interactions

The system is relatively straight-forward if the agents are capable of performing the task specified by their users. They simply query their own repositories, using their IR module, and return the retrieved documents to their user. This acts as a set of separate centralised IR systems.

The more complicated and interesting aspects of the system can be seen if an agent is unable to perform the task required (due to poor capabilities or repositories). If, for example, the agent in question does not have access to the information needed to fulfill its task(s) then it must employ the help of another agent. This is quite a complex task. In such a scenario agents must be able to communicate with one another, propose, reject, accept and counter propose courses of action. This situation may also occur on a larger scale. It is possible for an agent to 'employ' several agents to perform the task in question (coalition formation).

When an agent performs a task it expects a reward (modelled as an increase in user's trust in an agent). When a group of agents work together on a task they must also be rewarded for their efforts. This introduces the notion of a payoff scheme. The bonus available for the task is still the same regardless of how many agents work on the task. The bonus must be split in some way between the agents. Several coalitions may form within one agent community. There can be coalitions within coalitions. It should be noted that the agents are under no obligation to join in any coalition or even to cooperate with other agents.

In summary, we have a set of repositories, agents and users. These can be easily expanded if required. Primitives and protocols exist which allow retrieval, trading and coordination schemes (coalition formation). We envisage that this architecture can be re-used with different mechanisms in place in order to compare approaches.

5 Experimental Setup

5.1 Information Retrieval Systems

Information collections used in the system are standard IR test collections. A collection of abstracts from the Communications of the Association of Computing Machinery (CACM), the CISI collection and the Cranfield collection provide a substantial amount of documents for use in the system. Each agent in the system has the ability to search its own document collection. We require that different agents have different capabilities so we have several IR engines. The IR modules currently available in the system are: the Boolean model and the Vector Space model.

5.2 Multi-agent System Protocols

The main motivation behind the test-bed implementation is to allow various agent cooperation, coordination and coalition formation strategies to be easily examined. We hope to measure complexity (messages passed), fitness and user satisfaction.

An agent may want to form a coalition with another agent or set of agents if it is unable to perform the task required of it. In order to form a coalition the agent has to carry out the following steps:

- Select agent(s) to join the coalition based on belief and trust ratings obtained from previous interactions.
- Broadcast *propose* performatives to selected agent(s).
- Await responses (*accept, reject or counter-propose* performatives.)
- If the accepting agents are happy with the terms of the coalition then the contract is implemented. On the other hand if the agents are not satisfied with the terms of the contract then they will try to re-negotiate with the offering agent by counter-proposing. This cycle may have several iterations until the agents decide to join or to finally reject the coalition
- The contract is implemented. Each participating agent performs its given task.
- Finally the initiating agent modifies its belief and trust ratings of the participating agents based on their performance in the coalition.

Variations on the above can easily be added.

5.3 Experiments

Initial experiments involve the simulation of a distributed multi-user IR system wherein users information needs are satisfied via a set of information agents. A number of information repositories will be created and will be accessible by some or all of the agents. The goal again was twofold: firstly to conduct experiments regarding agents in an IR domain, and secondly to demonstrate its suitability as a framework for testing agent protocols.

Initial agent experiments will involve local updates of belief and trust models possessed by individual agents, experiments regarding coalitions and normative constraints can be easily incorporated into the existing system. The system, as it is designed, allows users, n agents, n information needs and m IR modules to interact within the agent community. The n agents satisfy the users by providing information, through the use of m different IR modules.

6 Results

6.1 Experiment 1 – Self-Belief

The first experiment involves monitoring changes in the agent's confidence over time. This is indicated by user satisfaction. This value is modified through simulated user feedback (this is achieved through the use of human relevance judgements which are provided with the collections) and changes after each retrieval. Initially the agents are assigned the value 0.5 (on a scale 0-1), to represent their beliefs. This suggests that they have neither a strong nor weak belief in their own ability. There are ten agents in the system, half of which perform vector-based retrievals, with the remainder performing Boolean retrievals. Figure 1 illustrates the average belief for the vector and Boolean agents after each retrieval is performed. Each agent performs twenty queries which are randomly selected.

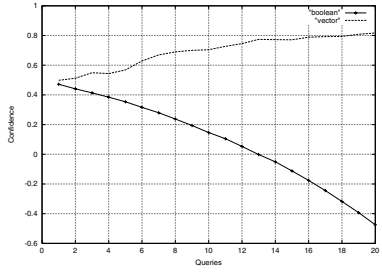


Fig. 1. Average performance of Boolean and vector agents

Figure 1 illustrates that the average user's trust in a vector-based agent is significantly higher than that of a Boolean agent. This indicates that the user has a higher confidence in, and is more satisfied with, vector-based agents. This allows us to posit that a Boolean agent will need to interact on a more frequent basis with other agents, preferably with vector agents, in order to satisfy its user.

6.2 Experiment 2 – Belief in Others

This experiment allows the agents to function as before but with the addition of the ability to modify their belief in other agent's abilities. This experiment

allows us to show that, over time, an agent can learn to trust or mistrust another agent. If an agent is unable to meet its user's information need, or it has experienced a decrease in its user's trust, it makes an attempt to 'employ' another agent to aid with a task. In order to select the agent to learn from, the agent examines its belief ratings. Initially these are set to 0.5, and the agent makes its choice randomly. The agent communicates with the selected agent, and asks it to perform the task. Again the selected agent will have no strong belief in any other agent, and it randomly decides whether or to accept or reject. Upon acceptance of the offer the selected agent searches its database and returns the results set to the 'employer' agent, who then returns the documents to the user as its own work. A successful interaction results in an increase in the agent's utility and its belief in the contracted agent.

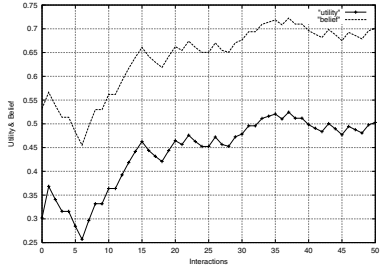


Fig. 2. Performance and belief over time

Figure 2 represents sample changes in self-belief (utility) and belief in other agents employed over time. As we can see this selected agent performs quite well over the first two queries and then deteriorates. When the 'employer' agent notices its belief in this agent decreasing, it selects another agent. Immediately we see a steady increase occur in both the utility and belief values. Upon examination of the agents involved we notice that the initial agent performs Boolean retrievals while the second agent is vector-based. We can see that agents learn over time to have greater trust in more reliable agents.

7 Summary and Conclusion

The areas of IR and multi-agent systems represent different areas of study but can be combined in order to provide a robust, extendible, intelligent, potentially distributed information management system. We believe that this approach provides a unique method of dealing with the traditional IR problem. The agents provide an intelligent module, in which they have the ability to communicate with, learn from, trust and distrust each other. We believe that this process of information retrieval provides a better quality of service to the end user. The multi-agent paradigm undoubtedly extends the capabilities of IR modules in a

distributed environment. We also posit that the system developed acts as a useful test-bed for experimentation regarding cooperation, coalition formation and negotiation. Immediate future work will involve the completion of experimentation using existing coordination and coalition schemes.

References

1. Sparck Jones, K., Willett, P.,(Eds.) 1997. *Readings in Information Retrieval.* Morgan Kauffman Publishers Inc. San Francisco, California.
2. Baeza-Yates, R., Ribeiro-Neto, B., 1999. *Modern Information Retrieval.* Addison-Wesley ACM Press, New York.
3. O'Riordan, C., Sorensen, H., 1999. *Information Retrieval and Filtering: An Overview,* in Dept of IT Journal, NUI, Galway.
4. Nwana, H., Ndumu, D., 1995. *An Introduction to Agent Technology,* in Journal of BT Labs.
5. Hewitt, C., 1997. *Viewing Control Structures as Patterns of Passing Messages,* in Artificial Intelligence.
6. Salton, G., Wang, A., Yang, C., 1975. *A Vector Space Model for Information Retrieval,* in Journal of the American Society for Information Science, volume 18, pages 613-620.
7. Huhns, M., Singh, M., (Eds.) 1998. *Readings in Agents.* Morgan Kaufmann Publishers Inc, San Francisco, California.
8. Weiss, G., 1998. *Learning to Coordinate Actions in Multi-Agent Systems,* in Huhns, et al. Readings in Agents.
9. Doran, J., Franklin, S., Jennings, N., Norman, T., 1997. *On Cooperation in multi-agent systems,* in The Knowledge Engineering Review, 12(3).
10. Wooldridge, M., Jennings, N., 1996. *Towards a Theory of Cooperative Problem Solving,* in Proceedings of Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-94).
11. Nwana, H., Lee, L., Jennings, N., 1996.*Coordination in Software Agent systems,* in Journal of BT Labs.
12. Shehory, O., Sycara, K., Jha, S., 1998. *Multi-Agent Coordination through Coalition Formation,* in Proceedings of Agent Theories Agent Languages and Mechanisms.
13. Zlotkin,G., Rosenschein, J., 1994. *Coalition, Cryptography and Stability: Mechanisms for Coalition Formation in Task Oriented Domains,* in Proceedings of the 12th National Conference on Artificial Intelligence.
14. Axelrod, R., 1984. *The Evolution of Cooperation.* Basic Books, New York.
15. Tuomela, R., 2000. *Cooperation: A Philosophical Study.* Philosophical Studies Series, Kluwer Academic Publishers.
16. Wooldridge, M., 1995. *Intelligent Agents: Theory and Practice,* in Knowledge Engineering Review.10(2).
17. Callan, James P., Zhihonh, Lu W., Croft, B., 1995. *Searching Distributed Collections with Inference Networks,* in Proceedings of the 18th Int. ACM SIGIR Conference on Research and Development in Information Retrieval
18. Voorhees, E., Siemens Trec-4 Report. *Further Experiments with Database Merging,* in Harman, D K., Proceedings of the 4th Text Retrieval Conference, 1998. pages 121-130.
19. Sparck Jones, K., Walker, S., Robertson, S., 1998. *A probabilistic model of information retrieval: development and comparative experiments,* in Information Processing and Management, 36(6).

All There Is to the Mind Is to Have the Right Genes, or, Consciousness as a Form of Genetic Engineering

Ajit Narayanan

Department of Computer Science, School of Engineering and Computer Sciences
University of Exeter, Exeter EX4 4PT, UK
A.Narayanan@ex.ac.uk

Abstract. Is there a gene or, rather, a set of genes which code for mind and consciousness? To refine this question further, can all our cognitive processes and states be accounted for mainly or even entirely in genetic terms? Clearly, environmental factors are important, since a person may have a gene for X which is never triggered because the right environmental conditions are not present. The question of whether there is a consciousness or mind gene can therefore be unpacked in two ways: whether, in principle, (a) there are systematic relationships of some sort between genes on the one hand and cognition/consciousness on the other, and (b) it is possible to give an account of conscious processes and states in genetic as well as environmental terms, assuming that such systematic relationships can be discovered. The aim of this paper is to explore the possibility that classical cognitive science will be superseded by biomolecular cognitive science in the near future as our understanding of the human genome grows and advances are made in gene chip technology.

1 Introduction

Genetic engineering (gene cloning) of products is now widespread. For instance, plants can be genetically engineered to produce their own insecticide by identifying single celled organisms, such as Bacillus thuringiensis, which have as part of their genetic make-up a gene or set of genes for producing insecticidal proteins (endotoxins) as a defence mechanism. If the organism is eaten by an insect, it releases the endotoxin which binds to the insect's gut and damages it to the point where the insect can starve to death, thereby stopping it from eating any more organisms. The precise part of the organism's genome for producing such endotoxins can be identified and cut out for insertion into crop seeds, such as maize or rice, so that the genes of the crop are supplemented by the organism's genes for producing insecticides (Fig. 1). If an insect then eats the crop grown from such genetically engineered seeds, it will suffer the same effects as if it had eaten the organism. Plant crop yields display significant improvements as a result of such genetic engineering. (A claimed beneficial side-effect is the reduced need for artificial pesticides.)

If important ethical considerations are put to one side for the moment, the vision being advocated here is that future philosophy and science of mind, consciousness studies and cognitive science (including computational neuroscience and artificial intelligence) will be influenced and possibly superseded by the possibility of the

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 78-86, 2002.

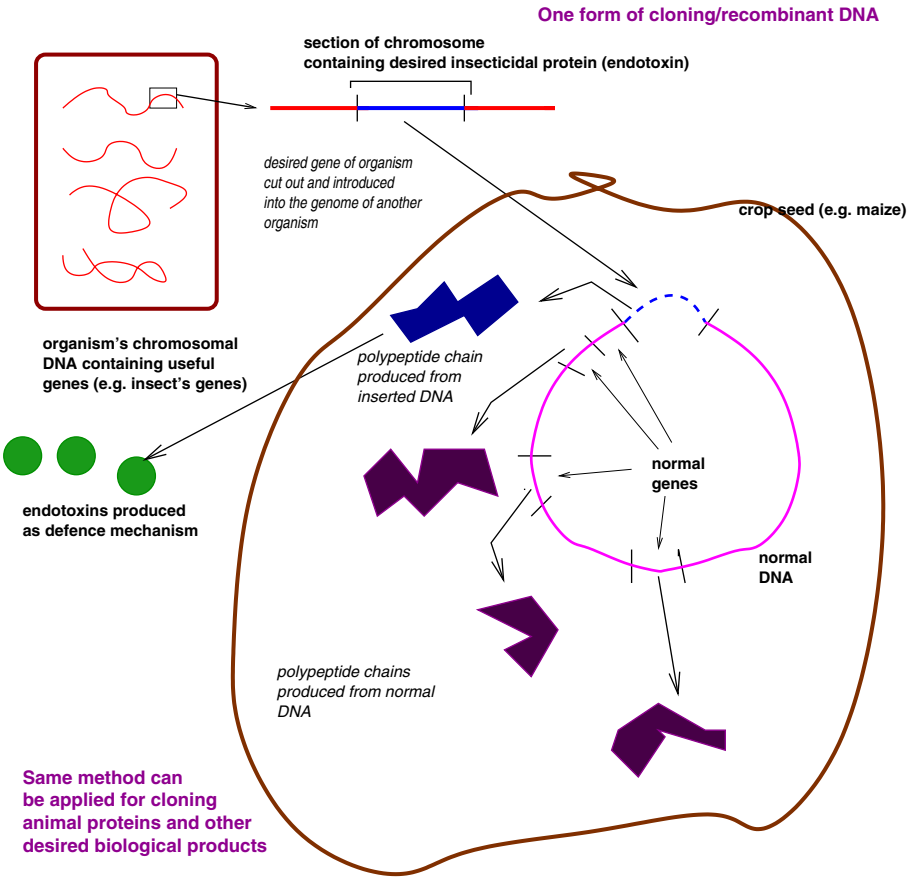


Fig. 1. Current gene cloning/recombinant DNA techniques typically find useful genes in one organism for use in another organism. For instance, if an organism naturally produces a certain endotoxin which acts as an insecticide, then the gene which produces that endotoxin, if identified in the chromosomal DNA of that organism, can be snipped out and inserted into the DNA of another organism, such as a crop seed, which will then produce the same endotoxin against insects 'naturally' when it grows into a multicellular plant (since all cells of the plant will contain the endotoxin gene introduced into the seed)

genetic engineering of intelligent entities by identifying human or animal genes associated with various cognitive or mental characteristics and then introducing them into the genomes of organisms, or artificially constructed biocellular automata, which do not possess such genes (Fig. 2).

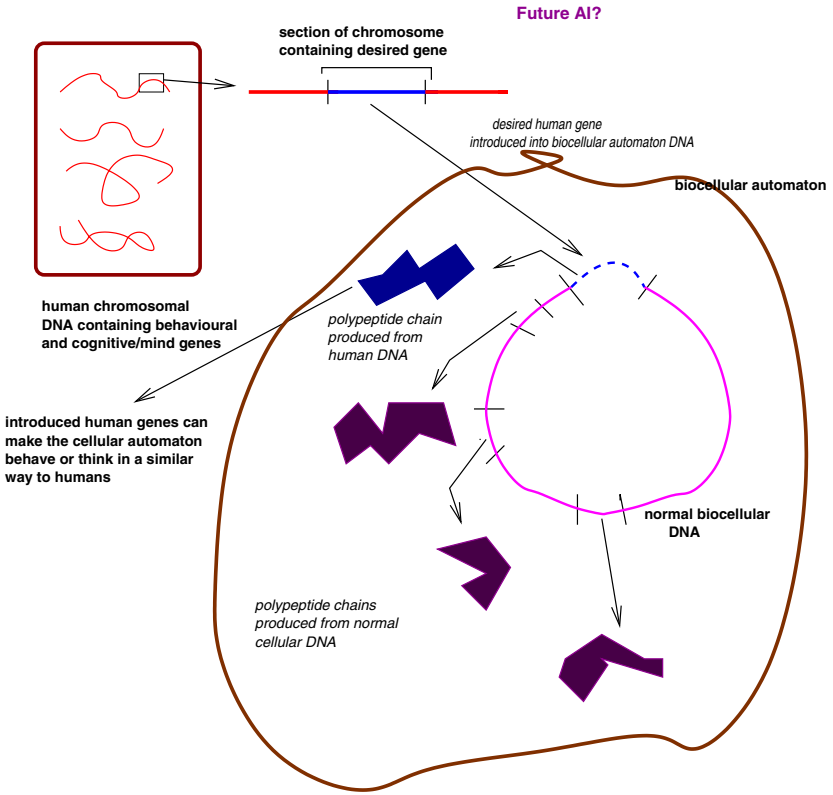


Fig. 2. Future mind/brain research may consist of waiting for biomolecular scientists to identify mind gene sequences among our chromosomes, isolating individual mind genes, introducing them into the DNA of artificial, single biocellular automata, and observing the results as polypeptide chains translated from the human DNA are produced.

Ethical constraints will, and should, prevent the use of existing, living organisms, but the technology exists currently for constructing simple artificial, biocellular automata which contain sufficient genes for their survival. It is quite possible that future mind/brain research will replace the symbolic programming and neural computing approaches with 'wetware platforms', i.e. collections of sophisticated artificial cells which can then be genetically engineered to produce desired mental and cognitive characteristics. In short, if there are human genes for consciousness, these could theoretically be inserted into the artificial cells' DNA, thereby conferring on the organism an innate ability to have consciousness. Just as one of the claims for cognitive science is that we have something which was not available to earlier philosophers and scientists to answer mind/brain questions, i.e. the notions of a digital computer and computation, biomolecular cognitive scientists can now claim that we have something which so far has not been available to earlier philosophers and scientists as well as current

cognitive scientists, i.e. an understanding of gene cloning and the human genome [1]. So much for the vision. The remainder of the paper addresses the question of how plausible this vision really is.

2 Scientific Context

There is a growing body of evidence to suggest that something like a biomolecular approach to cognition will emerge as the human genome is fully understood. Biomolecular and DNA-based accounts have already been provided of certain human aspects which have cognitive relevance, e.g. the perception of smells [2,3], tendencies towards aggressiveness and violence [4], and addiction [5]. There is also evidence that certain mental disorders have a genetic factor in addition to other factors (e.g. schizophrenia [6]) or have strong hereditary contributions (e.g. depression [7]).

DNA source accounts of mind, consciousness and cognition, to account for the variety and range of mental activity dependent on DNA processes, must hypothesise that numerous genes and their alleles play a part and that, furthermore, these genes are 'switched on' within neurons - neuronal genes. There will be different neuronal genes and their alleles for different aspects of cognition. The question of how fine-grained the relationship is between specific cognitive abilities and specific neuronal genes is an empirical one to be resolved by continuing research into the human genome, as is the precise location of these genes among the human chromosome set and the number of different alleles for a single mind gene.

There is early empirical evidence for genoneural architectures from work on genes encoding odour receptors [8]. An odour is first detected in the upper region of the nose where olfactory neurons pick up odour molecules via hairlike projections called cilia. The axons of these olfactory neurons extend from inside the nose to the brain through the thin bone which separates the olfactory bulb in the brain from the nasal cavity. Genes encoding odour receptor proteins are active only in olfactory neurons. More to the point, it was found that approximately 1,000 genes encode for 1,000 different odour receptors, with each type of receptor being expressed in thousands of olfactory neurons. Given that there may be as many as 50,000 genes in the human genome, this implies that about 2% of our genes are devoted to odour detection. The hypothesis is that each olfactory neuron expresses only one receptor gene ('one neuron, one receptor') [2,8]. Mammals can detect at least 10,000 odours because each of the 1000 different receptors responds to several odour molecules, with the brain then somehow determining the precise combination of activated receptors for particular odour identification.

This work exemplifies many aspects of what future genoneural architectures will consist of. First, the 'odour gene' is a collection of more specific genes for specific odours distributed across the human genome. Secondly, there is one gene for each odour receptor, and so the relation is one-to-one. This is to be contrasted with, say, visual receptors, where humans can distinguish among several hundred hues using only three kinds of receptor on the retina. The relationship here is one of overlap and differentiation. And thirdly, the perception of odours, including its identification and

re-identification, is caused by transcriptional and translational processes within olfactory neurons, where individual odour genes are switched on at the expense of other odour genes in the same neuron. The precise nature of the architecture which allows odour perception at the cognitive level is not currently known (a form of 'binding problem') but must be hypothesised to exist to account for how it is that we can smell certain odours and perhaps not others.

3 Implications for Research

Late 20th century research in consciousness science focused on the content and structure of cognition. Increasingly, correspondences between cognitive behaviour and brain activity are being discovered and recorded so that specific aspects of cognition are being identified with specific areas of the brain ('neural correlates of consciousness'). However, progress in these areas doesn't necessarily tell us how we can engineer entities which have consciousness and mentalism in a way which is somehow similar to ours. Giving a machine a program to process input and produce output using the very latest research in cognitive neuroscience doesn't overcome the objection that, at best, the machine simulates what we do. ('A simulation of a rainstorm doesn't leave us drenched,' Searle famously remarked.) Deeper explanations are required as to why localised brain activity is related to the cognitive aspect in question, and how this relationship is to be described and explained. The next stage in consciousness science research will be to look for source explanations of the correlations observed so that it will be possible to start engineering consciousness and cognition into basic 'living' entities which are alive precisely because they contain DNA. This will then provide an empirical way of determining which of the observed correlations between cognition and brain are accidental and which are systematic and causal.

There are two aspects to future, DNA-oriented cognitive science research. The first concerns how consciousness arose in the first place and addresses the architectural question. One way of tackling this question is to identify a small number of basic, artificial cell types, corresponding perhaps to the two basic types of primitive cell identified by many evolutionary biologists to be the source of all life on earth: prokaryotes and eukaryotes. That is, while artificial cells do not need to be identical with prokaryotes (no distinct nucleus containing the DNA which is stored as a ring) and eukaryotes (distinct nucleus containing DNA which is stored linearly), some basic types of cell architecture can be utilised as a starting point. For instance, every cell will require some transcription and translation mechanisms and modules. These artificial cells can be initially 'programmed' with DNA (through genetic engineering techniques) to ensure that they carry out fundamental biological activities to ensure survival and replication. Colonies of such cells can be allowed to evolve through methods understood well by genetic algorithm researchers: mutation, cross-over, adaptation and fitness. Once such colonies have evolved to the point where there is some stability in relation to their environment, further DNA can be added, again through genetic engineering, to promote intercellular interaction, through methods well understood by Alife researchers, such as communication with neighbouring cells and update of cell states depending on the states of such

neighbouring cells. Once basic intercellular interaction is demonstrated, the next stage will be to introduce, through genetic engineering, the types of intercellular interaction displayed by neurons so that structured architectures, such as feed-forward networks of cells, can evolve.

This process of architectural evolution will need to go hand in hand with a second type of research: the extraction and identification of gene networks responsible for mentalism and cognition. Microarray, or gene chip, technology [9] is developing at a rapid rate and allows researchers to identify which genes are being expressed at any particular moment within a cell under various conditions. Microarray data is created by observing gene expression values, or 'activation' values, over a number of time steps, often while subjecting the cell to some stimulus. The measurements gained from this study often span thousands of genes over a number of timesteps. The aim of analysing this data is to determine the pattern of activation and inhibition between genes which make up a gene regulatory network [10].

The sheer mass of such data has led many bioinformatics researchers to look for new computational methods to 'reverse engineer' the causal relationships between genes (when and why one gene expressed at one particular time point switches on or off, or regulates, other genes at a subsequent time point) [11] [12]. Assuming that the reverse engineering problem can be solved, microarray data from neurons operating under specified conditions can be used to extract gene networks to reveal and measure which neuronal genes are important for mentalism and cognition. These networks, or rather, the DNA for the genes involved in these networks, can then be introduced into evolved structured architectures to see whether aspects of mentalism and cognition are displayed as an emergent property of the low-level interaction between cells.

The above represents just one type of research strategy available to future, biomolecular cognitive scientists. Other types of research programme will also exist. For instance, there is already a widespread belief that gene expression data and gene regulation may be suserceded by proteomic data and protein-to-protein regulation. That is, there is growing evidence that one gene does not equal one protein, as the central dogma in molecular biology has so far assumed, but that one gene can give rise to many different types of protein. Gene expression data, by focusing on the products of gene transcription (into mRNA) rather than gene translation (into proteins), may not be telling us the whole story. It is possible that, just as cognitive science is currently split into symbolic and nonsymbolic approaches to cognition, biomolecular cognitive science may be equally split between DNA/gene researchers and proteomic researchers in terms of how best to measure cognition and mentalism as well as how best to represent it within cells.

4 Philosophical Implications

Given the experimental framework above, certain philosophical implications follow. The mind gene, or set of mind genes, if such exist, is that part of our genome (genotype) which represents a biocognitive genotype, where one biocognitive genotype is distinguished from another biocognitive genotype by the allelic values present in the

individual mind genes of the two biocognitive genotypes. The relation between, on the one hand, biocognitive genotypes and, on the other, biocognitive phenotypes which result from a combination of allelic values, while currently not known, is part of an intrinsic genocognitive architecture of the mind. Furthermore, when these genes are translated into proteins and when genes/proteins affect other genes/proteins through gene/protein networks, we have have biocognitive process which underlies what we, at the emergent level, call 'cognition'. All there is to the mind, according to this scenario, is to have the right genes.

This last statement needs qualification. Just as in AI Searle [13] characterised two positions, strong AI and weak AI, we can expect to see two schools of thought emerge in biomolecular cognitive science. The 'weak' position is that DNA accounts of mind give us a powerful tool when studying mind in that DNA accounts allow us to formulate and test hypotheses in a more rigorous and precise fashion than before. The equivalent weak biomolecular approach would be to say that, for example, gene networks, and the way that genes switch each other on and off, give us a systematic and formal way of testing whether the addition or removal of genes affects cognitive states. Such an approach would not deny that there may be something more to cognition than DNA processes, where this something extra may need explaining in terms different from DNA. A 'strong' position is that DNA accounts of mind, such as a gene network and the interactions of genes within this network, really do explain mind – there is nothing left to explain once we have the right gene network. According to this strong position, gene networks and gene interactions are both necessary and sufficient for explaining mind. That is, what counts as a thought or a thought process is really nothing more than the interaction of genes and proteins at the biomolecular level.

The 'strong' biomolecular position may seem strange to researchers brought up in the classical cognitive science tradition, for whom computational accounts of mind and cognition are axiomatic. However, such a strong biomolecular position is no stranger now than the computational position would have been in the 1950s to philosophers brought up in the behaviourist tradition. What is common to both a computational and biomolecular position is the attempt to move away from a dominant position which, after initial promise, has not returned the sort of success predicted at the outset. For instance, behaviourism ultimately failed to account for mind/brain because of its insistence on an 'whole organism' approach which did not consider mind to be relevant or, if relevant, not interesting to study in its own right. Similarly, computational accounts of mind/brain may also be said to have failed, given that despite over 50 years serious endeavour we are still no closer to a thinking, intelligent machine. The major successes in AI appear to be due to advances in hardware speed rather than software (program) advances.

To put this another way, there are four major questions in science. First, what are the conditions which gave rise to the big bang? This is a question for particle physicists and cosmologists. Second, given that there was a big bang, we can imagine a totally dead universe consisting of stars, galaxies and uninhabited planets. But, for some unknown reason, about 3 billion years ago on this planet something called 'life' emerged where, as far as we know, life is characterised by the use of DNA or RNA. What are the conditions which led to life starting? Third, we can imagine an Earth populated only by single celled creatures. About one billion years ago, something

remarkable happened: two or more single celled creatures, for one reason or another, formed a partnership to become a multicellular organism. Within a few hunded million years, we had gigantic multicelled creatures called dinosaurs. What are the conditions which gave rise to multicellular organisms? And fourth, although dinosaurs were in existence for a couple of hundred million years, there is no evidence that dinosaurs ever displayed intelligence in the form that we would understand. Dinosaurs didn't ever look into space, for instance, to determine whether they would be wiped out by a large meteorite, nor did they go to university, or create new games, languages, songs, plays, etc. But about 200,000 years ago, something happened to a multicellular ancestor of ours which resulted in Homo sapiens, and within a very short time (in evolutionary terms) we have what we call true intelligence or cognition. What are the conditions that give rise to intelligence?

The problem with computational cognitive science is that it deliberately pays no attention to the 'facts' of how intelligence arose or the empirical conditions which give rise to intelligence, leading some scientists to question whether cognitive science, and the discipline of computer science on which it is based, is a science at all, given the dependence of these sciences on mathematical foundations and conditions. The promise of biomolecular cognitive science is that it is firmly based in empirical fact: we are the only known entities who have intelligence, and any attempt to explain intelligence must be firmly rooted in the empirical conditions under which we operate. Biomolecular accounts of cognition and intelligence currently hypothesise that gene networks and protein-protein interactions provide the key to understanding what makes us similar as well as different from other life forms on this planet. This hypotheis may be wrong, but there is only one way to find out, and that is to undertake empirical research, such as the genetic engineering of consciousness.

References

1. Narayanan, A.: Biomolecular cognitive science. In: O Nuallain, S., Mc Kevitt, P., Mac Aogain, E. (Eds): Two Sciences of Mind: Readings in Cognitive Science and Consciousness, John Benjamins Publishing (1997) 21-36.
2. Buck, L., Axel, R.: A novel multigene family may encode odorant receptors: A molecular basis for odor reception. Cell 65:1 (1991) 175-187.
3. Mori, K., Yoshihara, Y.: Molecular recognition and olfactory processing in the mammalian olfactory system. Progress in Neurobiology 45:6 1995 585-619.
4. Raine, A.: The Psychopathology of Crime. Academic Press New York (1993).
5. Landry, D. W.: Immunotherapy for cocaine addiction. Scientific American (February 1997) 28-31.
6. Mirsky, A. F., Duncan, C. C.: Etiology and expression of schizophrenia: Neurobiological and psychosocial factors. Annual Review of Psychology 37 (1986) 291-321.
7. Moldin, S. O., Reich, T., Rice, J. P.: Current perspectives on the genetics of unipolar depression. Behavioural Genetics 21 (1991) 211-242.
8. Axel, R.: The molecular logic of smell. Scientific American (October 1995) 130-137.
9. Gershon, D: Microarray technology: An array of opportunities. Nature 416 (April 2002) 885-891.
10. Chen, T., Filkov, V., Skienna, S. S.: Identifying gene regulatory networks from experimental data. Third Annual International Conference on Computational Molecular Biology. ACM-SIGAT (2000) 94-103.

11. D'haeseleer, P., Wen, X., Michaels, G., Somogyi, R.: Mining the gene expression matrix: Inferring gene relationships from large scale gene expression data. Proceedings of the International Workshop on Information Processing in Cells and Tissues. Plenum Press New York (1998) 203-212.
12. Keedwell, E.C., Narayanan, A., Savic, D.: Modelling gene regulatory data using artificial neural networks. Proceedings of the 2002 International Joint Conference on Neural Networks (IJCNN'02) (2002), IEEE New Jersey.
13. Searle, J. R.: Minds, brains and programs. The Behavioural and Brain Sciences 3 (1980) 417-424.

Towards Robust Collaborative Filtering

Michael P. O'Mahony, Neil J. Hurley, and Guenole C.M. Silvestre

Department of Computer Science, University College Dublin
Belfield, Dublin 4, Ireland
{michael.p.omahony,neil.hurley,guenole.silvestre}@ucd.ie

Abstract. Collaborative filtering has now become a popular choice for reducing information overload. While many researchers have proposed and compared the performance of various collaborative filtering algorithms, one important performance measure has been omitted from the research to date. *Robustness* measures the power of an algorithm to make good predictions in the presence of erroneous data. In this paper, we argue that robustness is an important system characteristic, and that it must be considered from the point-of-view of potential attacks that could be made on the system by malicious users.

1 Introduction

The information overload problem has prompted much research and the collaborative filtering approach [2,8] has gained in popularity in recent years. Collaborative filtering attempts to filter information in a personalised manner, and operates by recommending items to a user based upon the transactions of similar users in the system. Such algorithms have now been implemented in many application areas, ranging from on-line book stores, movie and music finders, job recruitment services, etc.

Many approaches to collaborative filtering have appeared in the literature in recent times, and much work has been done in comparing the performance of the various algorithms. However, one important performance measure has been omitted from the research to date – that is the *robustness* of the algorithm. In essence, robustness measures the power of the algorithm to make good predictions in the presence of erroneous data. Generally, collaborative filtering applications update their datasets whenever users input new ratings, without any quality assurance that the entered rating is a true reflection of a real user's preference. Indeed, since rating entry can be an onerous process, users may be careless in the values that they enter and inaccuracies in the data must be expected. More sinisterly, it is possible to imagine scenarios in which malicious users may be motivated to deliberately attack the recommendation system and cause it to malfunction. Imagine, for example, publishers wishing to promote their work by forcing a book recommendation system to output artificially high ratings for their publications.

Most recommendation applications operate in a web environment in which it is impossible to check the honesty of those who access the system. Hence, it

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 87–94, 2002.

is important to understand how much tolerance the recommendation algorithm has to "noise" in the dataset. In this paper, we propose a definition for system robustness, and use the collaborative filtering algorithm described in [8] to identify system characteristics that influence robustness. Given that some modifications to this algorithm, such as inverse user frequency, case amplification, improved similarity measures etc [1] have been proposed from a predictive accuracy point-of-view, the effect of these enhancements upon robustness must also be investigated. In addition, several attack strategies are described in detail, and experimental results are presented for the various scenarios outlined.

2 Formal Framework

In our model of robustness, we take the end-user's perspective (to whom recommendations are delivered), and seek systems which are consistent in the recommendations that are made. We consider memory-based collaborative filtering (CF) in which the task is to predict the votes of a particular user (the *active* user) from a database of user votes, drawn from a population of other users. Let $\mathcal{U}$ be the universe of all users and fix some set of items I for which users vote. The votes of a user a on all items may be expressed as a vector, termed the *user profile* $\mathbf{v}_a \in V^m$ where V is a discrete set of vote values (including the special symbol $\perp$ interpreted as "not voted on") and m is the total number of items. A CF database D_U for $U \subset \mathcal{U}$ is a collection of votes $\{\mathbf{v}_a | a \in U\}$ for a particular set of users U.

Let $v_{i,j}$ be the j^{th} element of $\mathbf{v}_i$ corresponding to the vote for user i on item j. Using the notation of [1], define I_i as the set of items on which user i has voted. The predicted vote of the active user a for item j, $p_{a,j}$ is given by

$$p_{a,j} = \bar{v}_a + \kappa \sum_{i=1}^{n} w(a,i)(v_{i,j} - \bar{v}_i) \tag{1}$$

where $\bar{v}_i$ is the mean vote for user i, n is the number of users in the database with non-zero weights $w(a,i)$ and κ is a normalising factor.

The Pearson correlation coefficient weighting, proposed in [8] is adopted to explore and test our ideas on robustness in CF systems. Hence,

$$w(a,i) = \frac{\sum_j (v_{a,j} - \bar{v}_a)(v_{i,j} - \bar{v}_j)}{\sqrt{\sum_j (v_{a,j} - \bar{v}_a)^2 \sum_j (v_{i,j} - \bar{v}_i)^2}} \tag{2}$$

where the sum is over those items which have been voted for by *both* user a and user i. From now on, we refer to this algorithm as *Resnick's algorithm*.

In our implementation of Resnick's algorithm, we have imposed a limit to the number of users involved in the calculation of the predicted rating, i.e. a nearest-neighbour approach is employed with the top-k nearest users chosen on the basis of highest (absolute) correlation with the active user.

In this paper, we test our concepts of robustness using the above algorithm, and leave to future work the impact on robustness resulting from the various enhancements to the algorithm proposed in [1].

3 Definitions of Robustness

One of the underlying claims of knowledge-based systems (KBS) is that they can deal with incomplete, inaccurate or uncertain data. In this regard, the previous work of Hsu and Knoblock [4,5] and Groot et al [3], who propose methodologies for examining the robustness of knowledge discovery and knowledge-based systems, is of interest. However, much of the work done to date tends to focus on "normal" variations in the data. In this paper, we argue that robustness from the point-of-view of malicious attacks carried out on the system also needs to be considered.

We take the following approach to robustness of collaborative filtering. An *attack* is a transformation T which maps a database D_U to a new database $D'_{U'}$. Under the transformation, each vote in D_U $v_{i,j}$ is mapped to a new vote $v'_{i,j}$. If either $v_{i,j} = \perp$ or $v'_{i,j} = \perp$, this implies the addition of new votes or the deletion of existing votes, respectively. A transformation may also entail the addition of new users, so that $U \subseteq U'$.

Let $\mathcal{T}$ be the set of all possible attacks. Define a cost function $C : \mathcal{T} \to \mathcal{R}$. In general the cost of a transformation is application dependent. Various criteria can be used to construct a cost function. If the cost is related to the amount of effort it takes to perform the transformation, then this could be modelled by a monotonically increasing function of the number of user-item pairs which are modified by the transformation. There may also be a real cost, if ratings can only be entered into the system by purchasing the item in question.

We now propose definitions of robustness for a collarorative filtering system. Fix a set, A, of unrated user-item pairs in the database, which remain unrated in the transformed database i.e. $A \subseteq \{(a,j) | v_{a,j} = v'_{a,j} = \perp\}$. Robustness is defined over this set, A.

Definition 1. *For each user-item pair $(a,j) \in A$, the* prediction error (PE) *of prediction pre- and post-attack T is given by*

$$\mathrm{PE}(a,j,T) = p'_{a,j} - p_{a,j} \ . \tag{3}$$

Definition 2. *The robustness of the set A to an attack T is given by*

$$\mathrm{Robust}(A,T,\alpha) = 1 - \frac{1}{|A|} \sum_{a \in A} \kappa_{a,j}(\alpha) \tag{4}$$

where α is an arbritrary prediction shift. When $\alpha \geq 0$, $\kappa_{a,j}(\alpha) = 1$ if $\mathrm{PE}(a,j,T) \geq \alpha$, and 0 otherwise; when $\alpha < 0$, $\kappa_{a,j}(\alpha) = 1$ if $\mathrm{PE}(a,j,T) \leq \alpha$, and 0 otherwise. For any given α, a robustness value of 1 indicates no change in pre- and post-attck predictions (or an unsuccessful attack), while a value of 0 means that all predictions have been changed (successful attack). With this definition, a comprehensive picture of system robustness can easily be obtained by varying α and examining the trends that emerge. For example, a robustness value of 0.5 at $\alpha = 2$ indicates that 50% of all predictions were shifted by at least +2 units on a given rating scale.

Note that our definition of robustness is independent of the "true" rating for the user-item pair. This is an important feature of our scheme, because it abstracts system accuracy away from the concept of robustness.

4 Attacks

Various forms of attack are possible on CF systems, and in this section we discuss some potential scenarios that may occur in practice. In all cases, the strategy used to mount an attack is the same. An attack consists of a set of malicious user profiles, which is inserted (via the normal user-interface) into the database.

Note that with Resnick's algorithm, the attack profiles added need to be sufficiently close or similar to the target users if they are to have an influence on the predicted ratings. For example, consider an attack designed to promote a particular product. The ideal scenario would be that the attack profiles would correlate perfectly with the target users and therefore maximise the predicted rating of the product. Attack parameters, such as size and number of profiles added, along with the items and ratings that comprise each profile, will need to be adjusted to implement the desired attack. In the following sections, we discuss these aspects in detail and identify some system characteristics that may be exploited to reduce system robustness.

4.1 Random Attack

In a *random attack*, we consider that the goal of an attacker is to reduce the overall performance of a system as a whole. In these attacks, the focus is not on particular users or products, rather it is to target all users and items equally in an attempt to compromise the general integrity of the system. As a potential real-life scenario, consider a rival recommender system owner, who wishes to undermine the opposition and attract additional customers. The attack strategy in this case is relatively straightforward - the number of items in the attack profiles are randomly selected, along with the items and ratings that comprise them.

4.2 Product Push/Nuke Attack

A *product push* or *nuke* attack attempts to force the predicted ratings of a particular item, or group of items, to the maximum or minimum rating. In this attack, an important weakness in the Pearson correlation formula can be exploited. Since the correlation between users is calculated over only those items that *both* users have rated, this means that users can correlate strongly even though they have few items in common. While this weakness has been noted from the predictive accuracy point-of-view and some modifications have accordingly been proposed, we highlight here its implications for robustness of recommendation.

With Pearson's formula, the correlation between users who have *exactly* two items in common is always $+1$ or -1. If the attacker adopts a strategy of building

false user profiles consisting of the item to be pushed (or nuked), set at the maximum (or minimum) rating, together with two other carefully selected items, then the probability of a successful attack is high. Ideally, two items about which there is strong concencus in the user population are selected to generate the attack profiles.

The remaining difficulty in this strategy is in ensuring that the attack profiles correlate in the same way (i.e. either positively or negatively) with all the target users. To this end, some simple heuristics can suffice. It is a fair assumption that items that have many ratings in the database (i.e. popular items) are generally rated higher than average. Such items are therefore appropriate choices when building attack profiles.

5 Results

We used two different datasets to evaluate the attack scenarios outlined in the previous section. The `Movielens` dataset [6] consists of 943 users, 1,682 movies and contains 100,000 transactions in total. Movies are rated on a scale of 1 to 5. The second dataset, `PTV` [7], is more sparse and consists of 1,542 users, 8,129 TV programs, and has 58,594 transactions. The rating scale for this dataset is from 1 to 4. From experiment we set the number of nearest neighbours to 50 for all attacks involving both datasets. The experimental protocol adopted is one in which the test set is obtained by removing a single user-item pair from the dataset. and a prediction is then made for this pair using all the remaining data.

5.1 Experiment 1

In this experiment, a random attack is carried out on both datasets. Since the goal of this attack is to reduce the general robustness of both systems, we are concerned only with the absolute shift in predictions that occur, rather than with the direction (i.e. either positive or negative) of these shifts. Therefore, we take the absolute value of prediction shifts and use positive α values in our calculations using Eq. (4).

Figure 1 shows robustness trends plotted against α for 6 attacks of different strengths. In each attack, a percentage $X\%$ (expressed as a percentage of the original number of users in the datasets) of attack profiles are added. The number of items in each attack profile is a uniformly random number ranging between the minimum and maximum profile sizes that occur in the datasets. For both datasets, the attacks are generally unsuccessful, although `PTV` is more vulnerable. However, even here we see that `PTV` has a robustness of 0.9 against (absolute) prediction shifts of $\geq +1$ at $X = 500\%$ (i.e. when 5 times the number of original users are added as attack profiles). `Movielens` appears almost completely robust against these attacks, with robustness approaching 1 at low values of α.

These results can be explained by the fact that the distribution of randomly generated attack profiles is significantly different to the distribution of genuine users in the datasets, and therefore such attack profiles are unlikely to contribute to predictions. In addition, those attack profiles that do contribute to

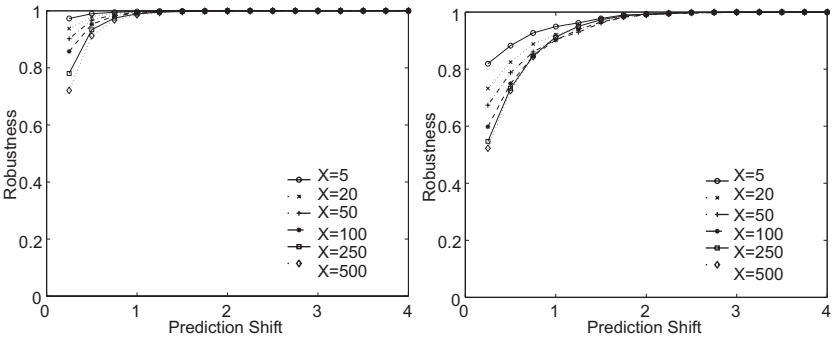


Fig. 1. Robustness for Movielens (left) and PTV (right) in Experiment 1

predictions are as likely to correlate positively as they are negatively with target users, thereby having no significant net effect. Thus robustness for both datasets remains high.

5.2 Experiment 2

This experiment shows the results of product nuke attacks on the datasets. As before, 6 attacks of different strengths are implemented. Since the goal of these attacks is to reduce predicted ratings, the direction of prediction shifts is of importance, and therefore both positive and negative values of α are used. In each attack, a percentage $X\%$ (expressed as a percentage of the original number of users in the datasets) of attack profiles are added. All of the attack profiles added are identical and each comprise 3 items: the item to be nuked together with the two most popular items in the datasets (as described in Sect. 4). Note that this attack can only be successful for those users who have rated the two most popular items (35% of all users for `Movielens` and 18% for `PTV`). Therefore, we measure robustness according to Eq. (4) over only those users who have rated the items in question.

Figure 2 presents robustness plotted against α for the datasets. Since in product nuke attacks the goal is to reduce predictions, negative prediction shifts correspond to a successful attack. For `Movielens`, the attacks are successful because large, negative prediction shifts predominate. For example, at $\alpha = -2$ and $X = 6.4\%$ (the strongest attack, where 60 attack profiles are added), robustness falls to 0.61, which means that 39% of predictions changed by at least -2. This is significant because in the `Movielens` rating scale of 1 to 5, this results in a prediction shift from "like" to "dislike". With respect to the positive prediction changes, these are relatively minor, with, for example, a robustness of 0.86 against shifts of $\geq +1$ at $X = 6.4\%$. For `PTV`, the attack is not successful, given that the magnitudes of the positive shifts are greater than those of the negative shifts, and with the exception of $\alpha = 0$, the minimum robustness achieved is still relatively high, at 0.7.

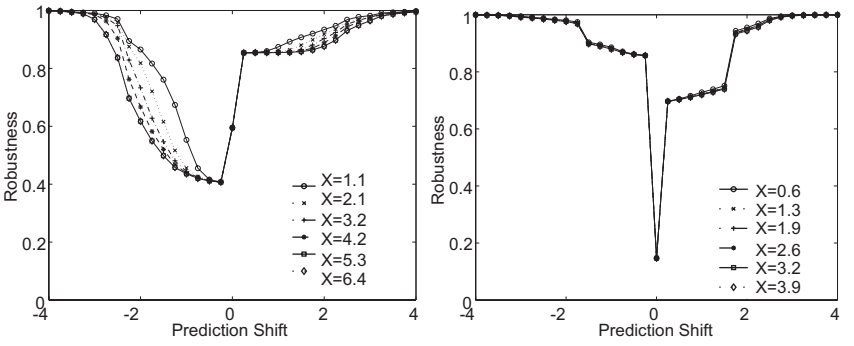


Fig. 2. Robustness for Movielens (left) and PTV (right) in Experiment 2

The high robustness values for the `PTV` dataset can be explained by the fact that the average number of attack profiles that contribute to predictions is significantly less for `PTV` than for `Movielens`. For example, in the strongest attacks, the average number of attack profiles that contribute to predictions is 73% for `Movielens`, as opposed to only 45% for `PTV`. Here we see the heuristic of assigning a higher rating to the most popular item in the attack profiles breaking down for `PTV`. This is due to the sparsity of the dataset, which is an order of magnitude more sparse than the `Movielens` dataset.

5.3 Experiment 3

It is clear that items which are rated identically over the entire dataset present a grave security risk from the point-of-view of robustness. This point is confirmed in this experiment, in which the `Movielens` dataset is initially modified so that two items have identical ratings over all users. These two items can now be used to generate very effective attack profiles to nuke (or push) any selected item in the dataset. We repeated the experiment outlined in Experiment 2 above on our modified dataset, and the results are presented in the form of a histogram (Fig. 3) of pre- and post-attack predictions, at $X = 6.4\%$. The histogram clearly illustrates the successful nature of the attack, where a very significant prediction shift occurs following the attack.

Note that although it may seem unlikely that a real dataset will contain such agreement across all users for any item, it is reasonable to expect that there are groups of users who will agree on some common sets of items. This experiment shows that if the goal of an attack is simply to target a particular user group, then a simple and effective strategy is to seek those items on which the group agree and use them to build attack profiles.

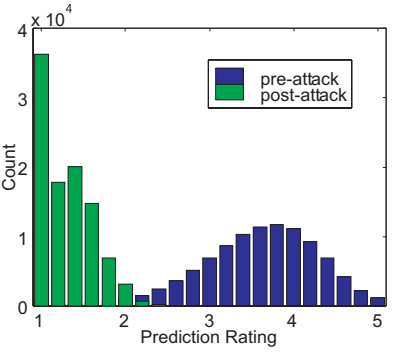


Fig. 3. Histogram of Pre- and Post-Attack Predictions in Experiment 3

6 Conclusion

In this paper we have introduced a new and important measure of performance, namely robustness, which has not previously been considered in the context of recommendation systems. We have demonstrated that a robustness analysis can highlight weaknesses in a system which do not become apparent by simply analysing recommendation accuracy. We have presented several attack scenarios and have shown that effective attack strategies can be devised to degrade the performance of the system. Future work will focus on robustness analysis of more sophisticated recommendation systems and on securing systems against attack.

References

1. J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 43–52, July 1998.
2. D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, December 1992.
3. P. Groot, F. van Harmelen, and A. ten Teije. Torture tests: a quantitative analysis for the robustness of knowledge-based systems. In *European Workshop on Knowledge Acquisition, Modelling and Management (EKAW'00)*. LNAI Springer-Verlag, October 2000.
4. C.-N. Hsu and C. A. Knoblock. Estimating the robustness of discovered knowledge. In *Proceedings of the First International Conference on Knwledge Discovery and Data Mining*, Montreal, Canada, 1995.
5. C.-N. Hsu and C. A. Knoblock. Discovering robust knowledge from dynamic closed-world data. In *Proceedings of AAAI'96*, 1996.
6. `http://movielens.umn.edu/`.
7. `http://www.changingworlds.com/`.
8. P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J.Riedl. An open architecture for collaborative filtering of netnews. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*. ACM, 1994.

GVR: A New Genetic Representation for the Vehicle Routing Problem

Francisco B. Pereira[1,2], Jorge Tavares[2], Penousal Machado[1,2], and Ernesto Costa[2]

[1]Instituto Superior de Engenharia de Coimbra, Quinta da Nora, 3030 Coimbra, Portugal
[2]Centro de Informática e Sistemas da Universidade de Coimbra, 3030 Coimbra, Portugal
{xico,machado,ernesto}@dei.uc.pt
jast@student.dei.uc.pt

Abstract. In this paper we analyse a new evolutionary approach to the vehicle routing problem. We present Genetic Vehicle Representation (GVR), a two-level representational scheme designed to deal in an effective way with all the information that candidate solutions must encode. Experimental results show that this method is both effective and robust, allowing the discovery of new best solutions for some well-known benchmarks.

1 Introduction

The Vehicle Routing Problem (VRP) is a complex combinatorial optimization problem, which can be seen as a merge of two well-known problems: the Traveling Salesperson (TSP) and the Bin Packing (BPP). It can be described as follows: given a fleet of vehicles with uniform capacity, a common depot, and several customer demands (represented as a collection of geographical scattered points), find the set of routes with overall minimum route cost which service all the demands. All the itineraries start and end at the depot and they must be designed in such a way that each customer is served only once and just by one vehicle. VRP is NP-hard, and therefore difficult to solve. Due to its theoretical and practical interest (it has numerous real world applications, given that distribution is a major part of logistics and a substantial cost for many companies), the VRP has received a great amount of attention since its proposal in the 1950's.

Due to the nature of the problem it is not viable to use exact methods for large instances of the VRP (for instances with few nodes, the branch and bound technique [1] is well suited and gives the best possible solution). Therefore, most approaches rely on heuristics that provide approximate solutions. Some specific methods have been developed to this problem (see, e. g., [2,3]). Another option is to apply standard optimization techniques, such as tabu search [4], simulated annealing [4,5], constraint programming [6], or ant colony optimization [7].

In the past few years there has also been some applications of evolutionary computation (EC) techniques to the VRP (for a good overview on this topic, consult [8]). Most researchers rely on hybrid approaches that combine the power of an EC algorithm with the use of specific heuristics (see, e.g., [9,10]) or use simplified versions of the problem. One common simplification is to pre-set the number of vehicles that is

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 95-102, 2002.

going to be used in the solution [11,12]. When applied alone, the success of EC techniques has been limited. In the literature we found two reports [13,14] about the application of non-specific EC methods to wide-ranging versions of this problem (i.e., versions that do not consider any kind of simplification). Both of them were tested in several well-known benchmarks and obtained results that are not very good, when compared to the best solutions found by other approaches.

We consider that the representation adopted for individuals plays a crucial role in the performance of an EC algorithm. In this paper we propose a new representational scheme intended to deal efficiently with the two levels of information that a solution must encode: clustering of the demands (i.e., allocation of all the demands to different vehicles) and specification of the delivery ordering for each one of the routes. This representation also enables an easy adjustment of the number of vehicles required for one possible solution. The search process relies on standard EC techniques. We do not use specific heuristics, nor do we perform any kind of simplification to the problem.

The paper has the following structure: in Sect. 2 we give a formal definition of the VRP. Section 3 comprises a description of the proposed EC model. In Sect. 4 we present and analyze the most important experimental results achieved. Finally, in Sect. 5, we draw some overall conclusions and suggest directions for future work.

2 The Vehicle Routing Problem

The most general version of the VRP is the Capacitated Vehicle Routing Problem (CVRP), which can be formally described in the following way: there is one central depot 0, which uses k independent delivery vehicles, with identical delivery capacity C, to service demands d_i from n customers, $i = 1, \ldots, n$. The vehicles must accomplish the delivery with a minimum total length cost, where the cost c_{ij} is the distance from customer i to customer j, with $i, j \in [1, n]$. The distance between customers is symmetric, i.e., $c_{ij}=c_{ji}$ and also $c_{ii}=0$. A solution for the CVRP would be a partition $\{R_1,\ldots,R_k\}$ of the n demands into k routes, each route R_q satisfying $\sum_{p \in R_q} d_p \leq C$. Associated with each R_q is a permutation of the demands belonging to it, specifying the delivery order of the vehicles. In Fig. 1 we present an illustration of the problem, viewed as a graph, where the nodes represent the customers.

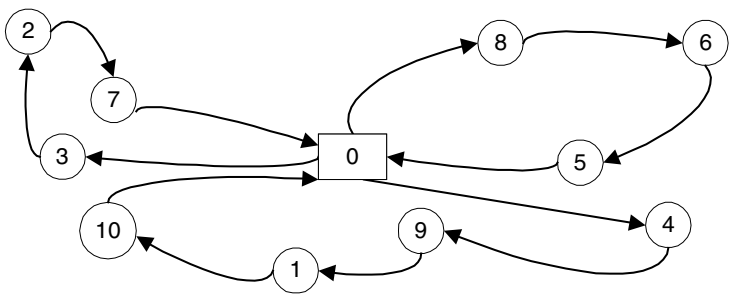


Fig. 1. Vehicle Routing Problem

3 Experimental Model

3.1 Genetic Vehicle Representation (GVR)

A candidate solution to an instance of the CVRP must specify the number of vehicles required, the partition of the demands through all these vehicles and also the delivery order for each route. We adopted a representation where the genetic material of an individual contains several routes, each one of them composed by an ordered subset of the customers. All demands belonging to the problem being solved must be present in one of the routes. As an example, the chromosome from Fig. 2 represents the solution presented in Fig. 1.

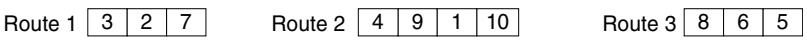


Fig. 2. An example of a GVR chromosome

Some routes in the chromosome may cause a vehicle to exceed its capacity. When this happens, and to guarantee that the interpretation yields a valid solution, we split the route that exceeds capacity in several ones. An example illustrates this adjustment: assume that the original route {a, b, c, d, e, f} causes the vehicle to exceed its capacity at node d. When this situation occurs, the itinerary is divided in two sections: {a, b, c} and {d, e, f}, and a new vehicle is added to the solution. If necessary, further divisions can be made in the second section. Notice that these changes only occur at the interpretation level and, therefore, the information codified in the chromosome is not altered.

3.2 Genetic Operators

The EC algorithm processes the individuals in a straightforward way. Assuming that the population size is N, in each generation N parents are chosen and N descendants are obtained through the application of genetic operators to the elements of the selected set.

We consider two categories of operators: crossover and mutation. They must be able to deal with the two levels of the representation. Thus, they should be capable to change the delivery order within a specific route and to modify the allocation of demands to vehicles. In this last situation, they can, not only switch customers from one route to another, but also modify the number of vehicles belonging to a solution (adding and removing routes). Another important requirement is that the genetic operators must always generate legal solutions.

The crossover operator used in our approach does not promote a mutual exchange of genetic material between two parents. Instead, when submitted to this kind of operation, one individual receives a fragment of genetic material (more precisely, a sub-route) from another parent and inserts it in one of its own routes. The donor is not modified. The geographical location of the customers is used to determine the position where the sub-route is inserted. The following algorithm clarifies how crossover is applied to the individuals of the selected set:

```
1. For each individual I1 from the selected set S Repeat:
   1.1. Randomly select another individual I2 from S
   1.2. From the genetic material of I2 randomly select a sub-route
        SR: {a1, a2, …, an}
   1.3. Find the customer c, not belonging to SR, that is geographi-
        cally closer to a1
   1.4. Insert SR in the genetic material of I1 in such a way that
        a1 is placed immediately after c
   1.5. From the original genetic material of I1 remove all dupli-
        cated customers that also appear on SR, obtaining a descen-
        dant D
```

The example from Fig. 3 helps to illustrate how crossover acts. It assumes that customer 6 is the one that is nearest to customer 9. This way, sub-route {9, 1, 10} is selected from I_2 and is inserted in one the routes of I_1, immediately after demand 6. The crossover operator makes possible that a fragment of information that is part of one individual can be incorporated into another solution. Given its behaviour, it cannot add new vehicles to the solution. On the other hand, it might remove routes.

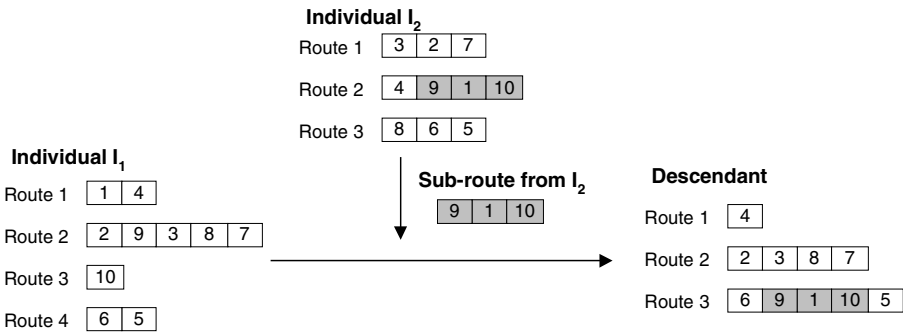


Fig. 3. Example of crossover

Descendants resulting from crossover can be subject to mutation. We consider 4 operators, based on proposals usually applied to order-based representations:

Swap: selects two customers and swaps them. Selected points can belong to the same or to different routes.

Inversion: selects a sub-route and reverses the visiting order of the customers belonging to it.

Insertion: selects a customer and inserts it in another place. The route where it is inserted is selected randomly. It is possible to create a new itinerary with this single customer. In all experiments reported in this paper, the probability of creating a new route is $1/(2\times V)$, where V represents the number of vehicles of the current solution. This way, the probability of creating a new route is inversely proportional to the number of vehicles already used. In Fig. 4 we show an example of this operation.

Displacement: selects a sub-route and inserts it in another place. This operator can perform intra or inter displacement (whether the selected fragment is inserted in the same or in another route). Just like in the previous operator, it is also possible to create a new route with the subsequence (the probability of this occurrence is calculated in the same way).

Swap and inversion do not change the number of routes of an individual. On the contrary, insertion and displacement have the ability to remove and to add vehicles to a solution. All genetic operators described have a specific probability of application.

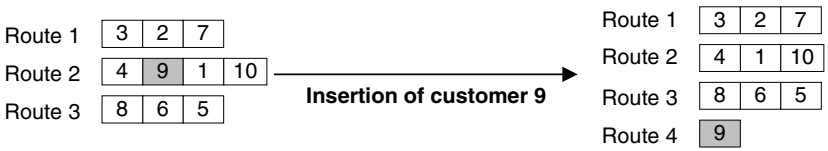


Fig. 4. Example of insertion mutation applied over customer 9

4 Experimental Results

To evaluate our approach we performed an extensive collection of tests with 12 instances from some well-known benchmarks: Augerat Set A (instances A32k5, A54k7, A60k9, A69k9, A80k10), Augerat Set B (instances B57k7, B63k10, B78k10) and Christofides and Eilon (instances E76k7, E76k8, E76k10, E76k14) [15].

The settings of the EC algorithm are the following: Number of generations: 50000 (except for the instance A32k5, where only 10000 generations were required); Population size: 200; Tournament selection with tourney size: 5; Elitist strategy; Crossover rate: 0.75; Mutation rates: swap: 0.05; inversion: {0.1, 0.15}; insertion: 0.05; displacement: {0.15, 0.2}. For every set of parameters we performed 30 runs with the same initial conditions and with different random seeds. All initial populations were randomly generated according to the following algorithm:

```
1. Build a set D with a random permutation of all the demands
2. While there are elements in D Repeat:
   2.1. Create one new route with K demands. K is a random value be-
        tween 1 and the current number of elements from D
   2.2. Remove first K elements from D and assign them to the new
        route
```

Even though values for different parameters were set heuristically, we performed some additional tests and verified that, within a moderate range, there was no significant difference in the outcomes.

In Table 1 we present, for all instances, the best solution found with each one of the 4 settings selected for this paper. Column “Nr. Points” identifies how many customers each instance has and column “Previous Best” summarizes the cost of the best solutions that were known when we started our research. A brief perusal of the results reveals that our EC approach was able to consistently find good solutions to the different instances used in the experiments. In six of them (A60k9, A69k9, B57k7, B78k10, E76k10 and E76k14), it discovered new best solutions with lower cost than those ones previously known. It is interesting to notice that in instance B57k7 the new solution found requires 8 vehicles, whereas the previous best required just 7 vehicles but had a higher overall cost (1153 vs. 1140). Since in the CVRP, the cost is the single objective to minimize, the solution we found is clearly better.

In Table 2 we show, for the same settings, the average of best solutions found in each one of the 30 runs. Columns labelled “Dist.” present the distance (in percentage) be-

tween these averages and the best-known solutions for the instances. Information from this table reveals that the representation used is very reliable. Considering the results as a whole, the distance that we just mentioned never exceeds 3.5% (in many situations, this value is considerably smaller). A detailed consult to both tables also shows that a small variation in the mutation rates of displacement and inversion does not produce major divergences in the results achieved. Other settings not presented here due to lack of space follow the same trend, showing that this method is robust: it was both able to find new best solutions for some of the instances used in the tests and to guarantee, with high likelihood, that good solutions are found during the search process.

Table 1. Best solutions found. Bold entries highlight new best solutions

Instances	Nr. Points	Previous Best	Inversion = 0.1		Inversion = 0.15	
			Disp = 0.15	Disp = 0.2	Disp = 0.15	Disp = 0.2
A32k5	32	784	784	784	784	784
A54k7	54	1167	1167	1167	1167	1167
A60k9	60	1358	1358	**1354**	1358	1358
A69k9	69	1167	**1159**	1165	1165	1165
A80k10	80	1764	1764	1777	1781	1774
B57k7	57	1153	**1140**	**1140**	**1140**	**1140**
B63k10	63	1496	1507	1496	1516	1497
B78k10	78	1266	1224	1223	1224	**1221**
E76k7	76	682	683	687	691	683
E76k8	76	735	737	738	738	740
E76k10	76	832	837	841	**830**	837
E76k14	76	1032	1028	**1022**	1031	1030

Table 2. Average of the best solutions found in each one of the 30 runs

In-stances	Inversion = 0.1				Inversion = 0.15			
	Disp = 0.15		Disp = 0.2		Disp = 0.15		Disp = 0.2	
	Avg.	Dist.	Avg.	Dist.	Avg.	Dist.	Avg.	Dist.
A32k5	793.4	1.2	790.9	0.9	796.5	1.6	786.3	0.3
A54k7	1178.6	1.0	1186.2	1.6	1182.4	1.3	1188.9	1.9
A60k9	1377.9	1.8	1377.9	1.8	1387.9	2.5	1372.4	1.4
A69k9	1180.6	1.9	1182.0	2.0	1180.0	1.8	1179.9	1.8
A80k10	1811.2	2.7	1813.8	2.9	1819.3	3.2	1811.0	2.7
B57k7	1141.1	0.1	1141.2	0.1	1141.7	0.1	1140.7	0.1
B63k10	1546.6	3.4	1544.0	3.2	1547.0	3.4	1545.7	3.3
B78k10	1255.3	2.8	1254.6	2.7	1249.5	2.3	1252.3	2.6
E76k7	705.9	3.5	704.8	3.3	703.4	3.1	705.0	3.4
E76k8	755.9	2.8	755.3	2.8	756.2	2.9	755.4	2.8
E76k10	851.7	2.5	856.6	3.2	852.6	2.7	854.0	2.9
E76k14	1042.9	2.0	1043.5	2.1	1045.2	2.3	1043.0	2.1

Another important feature of the algorithm is that it shows a good ability to escape from premature convergence to regions containing local optima. An example helps to illustrate this situation: we selected the instance with a higher number of customers (A80k10) and extended the search process to 100000 generations. Table 3 shows the effect of the duplication of the simulation time. With all setting, the average of the best solutions found in the 30 runs shows a slight decrease and, most important, it was

possible to find a new best solution with cost 1763. This example confirms that search is not stagnated. Even though it is possible to discover good solutions in early stages of the search process, if the algorithm is given enough time to carry on its exploration, it might continue to improve the best individuals found.

Table 3. Effect of the duplication of the simulation time with instance A80k10. The bold entry highlights a new best solution.

Nr. of generations	Inversion = 0.1				Inversion = 0.15			
	Disp = 0.15		Disp = 0.2		Disp = 0.15		Disp = 0.2	
	Best	Avg.	Best	Avg.	Best	Avg.	Best	Avg.
50000	1764	1811.2	1777	1813.8	1781	1819.3	1774	1811.0
100000	1764	1800.8	1777	1808.6	1775	1813.4	**1763**	1805.8

In this paper we will not perform a detailed analysis about the relative importance of each one of the 5 genetic operators used. This study will be carried out in a future publication. We want nevertheless to illustrate how crossover is important to discover good solutions. To achieve this goal, we performed an additional set of experiments that did not use this operator. Displacement mutation was selected to replace crossover. In Table 4 we present results of experiments performed with the following mutation rates: insertion and swap: 0.05; inversion: 0.15; displacement: {0.2, 0.5, 0.75}. All other settings remain unchanged. To help comparison, column "With Cx." shows the results achieved in the experiment performed with the following settings: crossover rate: 0.75; insertion and swap: 0.05; inversion: 0.15; displacement: 0.2. Even though this is just a preliminary study, a brief inspection of Table 4 reveals that the transfer of genetic material between parents seems to be crucial to the success of the proposed approach. Solutions found by experiments that used crossover are consistently better than all other situations that just relied on mutation. If we maintain our focus on the results presented on Table 4 and consider each one of the 12 instances separately, in most cases the average of the best solutions found by the experiment that used crossover is significantly lower (significance level: 0.05) than those ones achieved by experiments that just relied in mutation. The only exceptions occur in the 3 experiments performed with instance B57k7 and in one of the experiments done with instance A57k7 (the one that used displacement rate = 0.5).

Table 4. Summary of the influence of crossover on the search process

Instances	Disp = 0.2		Disp = 0.5		Disp = 0.75		With Cx.	
	Best	Avg.	Best	Avg.	Best	Avg.	Best	Avg.
A32k5	784	816.3	784	809.1	784	813.5	784	786.3
A54k7	1176	1219.3	1167	1199.4	1172	1209.4	1167	1188.9
A60k9	1363	1422.5	1358	1414.2	1377	1415.7	1358	1372.4
A69k9	1175	1213.7	1165	1199.6	1172	1209.7	1165	1179.9
A80k10	1801	1783.5	1813	1868.5	1851	1897.0	1174	1811.0
B57k7	1140	1142.7	1140	1141.2	1140	1142.2	1140	1140.7
B63k10	1507	1569.9	1504	1558.4	1548	1571.1	1497	1545.7
B78k10	1253	1289.6	1266	1289.4	1243	1281.4	1221	1252.3
E76k7	692	718.4	692	713.3	692	722.5	683	705.0
E76k8	747	778.3	748	768.7	738	767.4	740	755.4
E76k10	861	881.1	846	866.1	846	869.0	837	854.0
E76k14	1040	1062.9	1044	1064.3	1040	1064.4	1030	1043.0

5 Conclusions and Further Work

In this paper we presented a new generic evolutionary approach to the VRP. The two-level representational scheme we proposed proved to be extremely effective to this problem, since we were able to find new best solutions for several instances belonging to well-known benchmarks. Moreover, results show that this method is both robust and scalable.

Results presented here can be considered as preliminary. As future work we intend to perform a detailed study on the importance of the genetic operators presented in this paper and also extend our approach to other variants of the problem, such as the VRP with time windows.

References

1. Desrosiers, J., Madsen, O., Solomon, M. and Soumis, F. (1999). 2-Path Cuts for the Vehicle Routing Problem with Time Windows, *Transportation Science*, Vol. 33, No. 1, pp. 101-116.
2. Thangiah, S., Potvin, J. and Sun, T. (1996). Heuristic Approaches to Vehicle Routing with Backhauls and Time Windows, *Int. Journal of Computers and Operations Research*, pp. 1043-1057.
3. Prosser, P. and Shaw, P. (1997). Study of Greedy Search with Multiple Improvement Heuristics for Vehicle Routing Problems, *Technical Report RR/96/201*, Department of Computer Science, University of Strathclyde, Glasgow.
4. Tan, K. C., Lee, L. H., Zhu, Q. L. and Ou K. (2000). Heuristic Methods for Vehicle Routing Problem with Time Windows, *Artificial Intelligent in Engineering*, pp. 281-295.
5. Bent, R. and Hentenryck, P. V. (2001). A Two-Stage Hybrid Local Search for the Vehicle Routing Problem with Time Windows, *Technical Report, CS-01-06*, Brown University.
6. Shaw, P. (1998). Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems, *Proceedings of the Fourth International Conference on Principles and Practice of Constraint Programming (CP '98)*, M. Maher and J. Puget (eds.), pp. 417-431.
7. Gambardella, L. M., Taillard, E. and Agazzi, G. (1999) MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows, In D. Corne, M. Dorigo and F. Glover (eds.), *New Ideas in Optimization*. McGraw-Hill, London, UK, pp. 63-76.
8. Bräysy, O. (2001). Genetic Algorithm for the Vehicle Routing Problem with Time Windows, Arpakannus 1/2001, *Special Issue on Bioinformatics and Genetic Algorithms*, pp.33-38.
9. Thangiah, S. R. (1995). Vehicle Routing with Time Windows Using Genetic Algorithms, *Application Handbook of Genetic Algorithms: New Frontiers*, Volume II. Chambers, L.(ed), pp. 253-277, CRC Press.
10. Potvin, J. , Dubé, D. and Robillard, C. (1996). Hybrid Approach to Vehicle Routing Using Neural Networks and Genetic Algorithms, *Applied Intelligence*, Vol. 6, No. 3, pp. 241-252.
11. Zhu, K. (2000). A New Genetic Algorithm for VRPTW, *International Conference on Artificial Intelligence*, Las Vegas, USA.
12. Louis, S. J., Yin, X. and Yuan, Z. Y. (1999). Multiple Vehicle Routing With Time Windows Using Genetic Algorithms, *Proceedings of the Congress of Evolutionary Computation (CEC-99)*, pp. 1804-1808.
13. Duncan, T. (1995). Experiments in the Use of Neighbourhood Search Techniques for Vehicle Routing. *Report AIAI-TR-176*, University of Edinburgh.
14. Machado, P., Tavares, J., Pereira, F. B. and Costa, E. (2002). Vehicle Routing Problem: Doing it the Evolutionary Way, *To appear at GECCO-2002 Proceedings*.
15. Available at: `http://www.branchandcut.org/VRP/data/`.

An Empirical Comparison of Particle Swarm and Predator Prey Optimisation

Arlindo Silva[1,3], Ana Neves[1,3], and Ernesto Costa[2,3]

[1] Escola Superior de Tecnologia, Instituto Politécnico de Castelo Branco,
Av. do Empresário, 6000 Castelo Branco – Portugal
{arlindo,dorian}@est.ipcb.pt
[2] Departamento de Engenharia Informática, Universidade de Coimbra,
Pólo II – Pinhal de Marrocos, 3030 Coimbra – Portugal
ernesto@dei.uc.pt
[3] Centro de Informática e Sistemas da Universidade de Coimbra,
Pólo II - Pinhal de Marrocos, 3030 Coimbra - Portugal

Abstract. In this paper we present and discuss the results of experimentally comparing the performance of several variants of the standard swarm particle optimiser and a new approach to swarm based optimisation. The new algorithm, which we call predator prey optimiser, combines the ideas of particle swarm optimisation with a predator prey inspired strategy, which is used to maintain diversity in the swarm and preventing premature convergence to local suboptima. This algorithm and the most common variants of the particle swarm optimisers are tested in a set of multimodal functions commonly used as benchmark optimisation problems in evolutionary computation.

1 Introduction

The particle swarm algorithm is a new evolutionary algorithm, where the population is now called a swarm and each individual is called a particle [1]. It was inspired by the dynamics of social interactions between individuals, being initially influenced by work in simulation of coordinated flight in flocks of birds. This algorithm, which is frequently called Particle Swarm Optimiser (PSO), has been successfully used as an alternative to other evolutionary techniques in the optimisation of n-dimensional real functions.

The PSO is usually considered a form of evolutionary computation, but it does not make use of mutation or recombination operators and even selection does not exist in an explicit way. Instead, particles move in a coordinated way through the n-dimensional search space towards the function optimum. Their movement is influenced not only by each particle own previous experience, but also by a social compulsion to move towards the best position found by its neighbours. To implement these behaviours, each particle is defined by its position and velocity in the search space. In each iteration, changes resulting from both influences in the particle's trajectory are made to its velocity. The particle's position is then updated accordingly to the calcu-

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 103-110, 2002.

lated velocity. The PSO, its main variants and the cultural model behind it are extensively discussed in [2].

While the PSO has revealed itself capable of competing with other evolutionary techniques, namely the standard genetic algorithm (GA), it has been noted [3] that the original PSO had difficulties controlling the balance between exploration (global investigation of the search place) and exploitation (the fine search around a local optimum). According to [3], the PSO, while quickly converging towards an optimum in the first iterations, has problems when it comes to reach a near optimal solution. To solve this, [4] introduced the use of a linear decreasing inertia weight, reminiscent from the temperature parameter in simulated annealing. In [5] results of using the PSO with the inertia weight on several benchmark functions are presented and is concluded that, while performing significantly better that the original PSO, lacks global search ability at the end of the run. A second approach to controlling convergence in PSO introduces the use of a constriction coefficient [6].

We present here a new approach to balancing exploration and exploitation in PSO, by introducing a second population of particles, which we call predators. Predators have a different dynamic behaviour from the swarm particles (which we call prey). They are attracted to the best individuals in the swarm, while the other particles are repelled by their presence. Controlling the strength and frequency of the interactions between predators and prey, we can influence the balance between exploration and exploitation and maintain some diversity in the population, even when it is approaching convergence, thus reducing the risk of convergence to local sub-optima. We also present and discuss the experimental results obtained by comparing this model with the inertial weight and the constriction coefficient PSO variants in a set of benchmark functions.

The remainder of is article is organized in the following way: In the next section we briefly describe the standard PSO model and, in more detail, the new predator-prey optimiser (PPO). Description of the experimental setup and presentation of the results obtained is made in section 3. Finally, section 4 is dedicated to the presentation of some conclusions and objectives for future work.

2 The Optimiser Algorithms

2.1 The Swarm Particle Optimiser

In particle swarm optimisation a population of point particles “fly” in an n-dimensional real number search space, where each dimension corresponds to a parameter in a function being optimised. The position of the particle in the search space is represented by a vector X. The velocity of the particle, i.e., its change in position, is represented by a vector V. The particle “flies” in the search space by adding the velocity vector to its position vector in order to change its position.

V determines the particle’s trajectory and depends on two “urges” for each particle i: flying towards its best previous position and flying towards its neighbours’ best previous position. Different neighbourhood definitions have been tried [7]; here we assume that every particle is a neighbour to every other particle in the swarm. The

general equations for updating the position and velocity for some particle i are the following:

$$\begin{cases} V_i(t) = \chi(wV_i(t-1) + \varphi_{1i}(P_i - X_i(t-1)) + \varphi_{2i}(P_g - X_i(t-1))) \\ X_i(t) = X_i(t-1) + V_i(t) \end{cases} \quad (1)$$

In the above formula χ is the constriction coefficient described in [6], φ_1 and φ_2 are random numbers distributed between 0 and an upper limit and different for each dimension in each individual, P_i is the best position particle i has found in the search space and g is the index of the best individual in the neighbourhood. The velocity is usually limited in absolute value to a predefined maximum, V_{max}. The parameter w is a linear decreasing weight. The swarm is usually run for a limit number of iterations or until an error criterion is met.

From (1) we can derive the two most usual ways in which convergence and, as a result, the balance between exploration and exploitation are controlled. [5] uses χ=1 and weight w decreasing linearly from w_{max} to w_{min} during the execution of the algorithm. In [6] convergence is guarantied by choosing appropriated values for χ and $\varphi=\varphi_1+\varphi_2$. w is fixed and equal to 1 in this approach.

2.2 The Predator Prey Optimiser

Our motivation for developing the predator-prey model was mainly to introduce a mechanism for creating diversity in the swarm at any moment during the run of the algorithm, not depending on the level of convergence already achieved. This would allow the "escape" of particles even when convergence of the swarm around a local sub-optimum had already occurred. A second, and less practical, motive was to maintain the swarm intelligence principle behind the algorithm. Other mechanisms could perhaps have been used to the same effect, but it seemed more appropriate to introduce a mechanism that could also be implemented as a distributed behaviour in the swarm. The predator-prey model is inspired in the hunt of animals grouped in flocks by one or more predators. When chased, animals have more difficulty to stay around their most preferable places (better pastures, water sources…) and have to search for other locations, free of predators and perhaps even better. This is the effect we want to model in our algorithm, where the metaphorical better pastures are the functions' local sub-optima.

In the present state of development of the predator-prey optimiser, only one predator is used. The predator's objective is to pursue the best individual in the swarm, i.e. the individual that has found the best point in the search space corresponding to the function being optimised. The predator update equations are:

$$\begin{cases} V_p(t) = \varphi_4(X_g(t-1) - X_p(t-1)) \\ X_p(t) = X_p(t-1) + V_p(t) \end{cases} \quad (2)$$

φ_4 is another random number distributed between 0 and an upper limit and X_g is the present position of the best particle in the swarm. The upper limit on φ_4 allows us to control how fast the predator "catches" the best individual.

The influence of the predator on any individual in the swarm is controlled by a "fear" probability P_f, which is the probability of a particle changing its velocity in one of the available dimensions due to the presence of the predator. For some particle *i*, if there is no change in the velocity in a dimension *j* the update rules in that dimension still are:

$$\begin{cases} v_{ij}(t) = wv_{ij}(t-1) + \varphi_{1ij}(p_{ij} - x_{ij}(t-1)) + \varphi_{2ij}(p_{gj} - x_{ij}(t-1)) \\ x_{ij}(t) = x_{ij}(t-1) + v_{ij}(t) \end{cases} \quad (3)$$

But if the predator influences the velocity in dimension *j*, the rule becomes:

$$\begin{cases} v_{ij}(t) = wv_{ij}(t-1) + \varphi_{1ij}(p_{ij} - x_{ij}(t-1)) + \varphi_{2ij}(p_{gj} - x_{ij}(t-1)) + \varphi_{3ij}D(d) \\ x_{ij}(t) = x_{ij}(t-1) + v_{ij}(t) \end{cases} \quad (4)$$

The fourth term in the first equation in (4) quantifies the repulsive influence of the predator by modifying the velocity adding a value that is a function of the difference between the position of the predator and the particle. *d* is the Euclidean distance between predator and prey. *D(x)* is an exponential decreasing distance function:

$$D(x) = ae^{-bx} \quad (5)$$

D(x) makes the influence of the predator grow exponentially with proximity. The objective of its use is to introduce more perturbation in the swarm when the particles are nearer the predator, which usually happens when convergence occurs. When the distance is bigger (e.g. during the initial exploration phase of the swarm, when *w* is still big), the predator's influence is smaller and usual swarm dynamics take control. The *a* and *b* parameters define the form of the *D* function: *a* represents the maximum amplitude of the predator effect over a prey and *b* allows to control the distance at which the effect is still significant.

The predator effect was designed to take advantage of the use of *w* as an inertia parameter in the swarm update equations. The idea is to lower the values of *w*, thus forcing a faster convergence, while relying on the predator to maintain population diversity. The constriction coefficient is set to 1.

3 Experimental Results

3.1 The Experimental Setup

We tested the performance of the predator-prey model in four non-linear functions, all commonly used as benchmarks for evolutionary algorithms (e.g. [8]). The first two are also frequently used as benchmarks in swarm particle literature [2,3,5].

$$
\begin{aligned}
f_1(x) &= \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10) \\
f_2(x) &= \frac{1}{4000}\sum_{i=1}^{n} (x_i - 100)^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i - 100}{\sqrt{i}}\right) + 1 \\
f_3(x) &= -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e \\
f_4(x) &= 418.9829n + \sum_{i=1}^{n} -x_i \sin\left(\sqrt{|x_i|}\right)
\end{aligned}
\tag{6}
$$

In the functions above, x is a real number, n-dimensional vector and x_i is the ith element of that vector. f_1 is the generalized Rastrigin function, f_2 is the generalized Griewank function and f_3 the generalized Ackley function, three multimodal functions with many local minima set around the global minima in an unimodal macrostructure. f_4 is the Schwefel function which is a multimodal function, designed to be difficult for evolutionary algorithms, with the local minima set far away from each other and the global minimum located at the boundary of the search space. All these functions are used as minimization problems.

In our first set of experiments the predator prey algorithm was used to optimise the above functions. To implement the objective mentioned before of using the weight limits to force a faster convergence, with the predator-prey effect being expected to provide the lacking population diversity and avoid convergence to sub-optima, the weight limits were set to 0.5 and 0.0, far from the usual 0.9 and 0.4. φ_1 and φ_2 were set to 2. The a, b, and P_f values were, for each function, empirically and after a small set of preliminary experiments, set to the values presented in Table 1. No systematic effort was made to determine optimum values for these parameters.

Table 1. Search space and initialisation limits for each function, as well as parameter values used for the predator prey algorithm

Function	Search Space	Init. Limits	a	b	P_f
f_1	$[-10,10]^n$	$[2.56,5.12]^n$	$0.1X_{max}$	$10.0/X_{max}$	0.0005
f_2	$[-600,600]^n$	$[300,600]^n$	$0.1X_{max}$	$10.0/X_{max}$	0.002
f_3	$[-30,30]^n$	$[10,20]^n$	$2.0X_{max}$	$10.0/X_{max}$	0.0005
f_4	$[-500,500]^n$	$[-500,500]^n$	$0.1X_{max}$	$10.0/X_{max}$	0.001

We also ran three sets of experiments with variants of the standard particle swarm algorithm. PSOa corresponds to the PSO with the constriction coefficient set to one and a weight w decreasing linearly between 0.9 and 0.4, values recommended in PSO literature [4]. PSOb is similar to the PSOa but w decreases from 0.5 to 0.0 to allow a direct comparison with the PPO, which uses the same values. PSOc is a constricted version, with χ=0.739 and φ=4.1, values used in [9]. In all experimental settings V_{max} was set to the same value as X_{max}.

All experiments were run with functions having 50 dimensions, an iteration limit of 5000 and a population size of 20 particles. Care was taken to initialise the population in an asymmetric way for all functions except for f4, which, since its structure is not symmetrical around a local minimum situated near the origin, does not benefit from a symmetric initialisation. Table 1 presents the search space and initialisation limits for each of the test functions. For each experimental setting 100 runs of the algorithm were performed.

3.2 Results and Discussion

Table 2 presents the results obtained in the sets of experiments described in the previous section. For each experimental setting we present a 90% confidence interval for the average best solution found in the limit number of iterations, over the 100 runs. Figure 1 presents graphs illustrating the evolution of average fitness over the run for the four benchmark functions.

Table 2. Confidence intervals at 90% for the average best solutions of each experimental setting over 100 runs

F	PPO	PSOa	PSOb	PSOc
f_1	5.9357±0.8802	105.8131±4.8350	197.0707±8.1454	282.1091±11.7559
f_2	0.0080±0.0022	0.0136±0.0033	0.0372±0.0145	0.1810±0.0657
f_3	9.1955E-09±2.0367E-09	0.3460±0.1255	2.0451±0.1918	8.0508±0.5647
f_4	797.6098±57.9868	4167.4452±134.8793	6213.0122±156.4541	8242.2676±178.2687

We start by comparing the performances of the three PSO based approaches. It is easy to see from Table 2 that the particle swarm optimiser (PSOa) with weight varying from 0.9 to 0.4 is the one that performs better in all experimental settings. From Fig. 1 can be concluded that this may be caused by the faster convergence of the two other variants in the initial iterations, which causes a decrease in diversity resulting in early sob-optimal convergence for the PSOb and PSOc. It is interesting to compare these results with the ones in [9] where PSOc performed better than PSOa in different experimental settings: of the four functions used only two were common with ours, the functions were optimised in 30 dimensions and, more importantly, the result of each run was the number of iterations the algorithm took to find a solution within a predefined error form the global optimum. Since the error criterion was rather loose, the faster convergence of the PSOc made it perform better than the PSOa in those conditions. But, from our experiments, it seems that when we ask for finer convergence the weight decreasing version of the PSO easily outperforms the constricted version.

In a function to function qualitative analysis, and since the global optimums all have 0.0 fitness, we can say that the PSOa performed well in f_2 and f_3, not very good in the Rastrigin function and rather poorly in f_6 where escaping local minima is harder than in the other functions, since local optima are far apart in the search space. We can add from Fig. 1 that the PSOa fitness curves seems stabilized at the iteration limit (the same happens with PSOb and PSOc) and doesn't seem probable that an increment in fitness would result from a higher iteration limit.

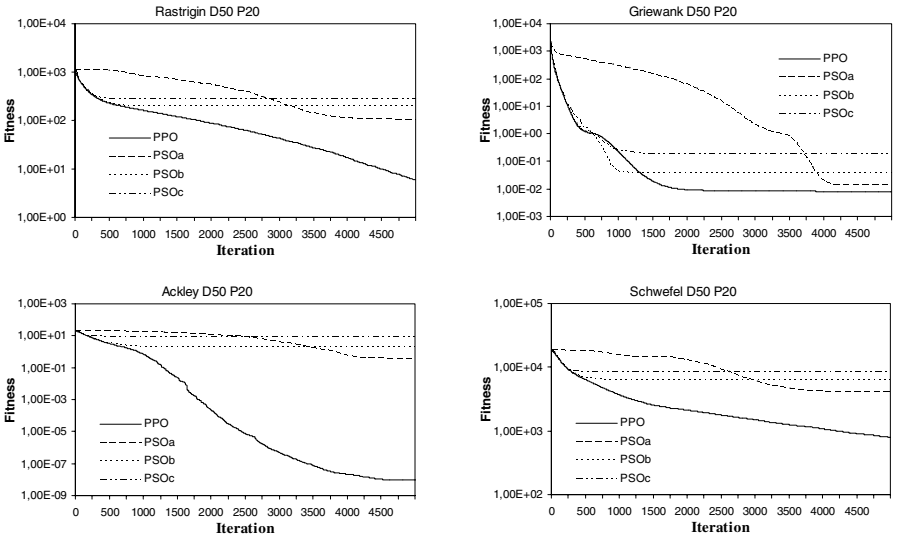


Fig. 1. Graphs illustrating evolution of average fitness of PPO versus PSO approaches for the four test functions. Fitness is presented in logarithmic scale

The experimental results using the predator prey approach allow us to conclude that the PPO performed significantly better (according to t-tests with $p<0.05$ criterium) than the best PSO approach (PSOa). As we were expecting, the extra diversity introduced by the predator prey mechanism allowed us to use a shorter "cooling schedule" for the linear decreasing weight which resulted, as can be seen in Fig. 1, in an initial convergence as fast for the PPO as for the PSOb and PSOc. But, while these two algorithms get easily caught in local minima, with their fitness curves stabilising very early in the runs, the PPO keeps getting better solutions until the iteration limit is reached, with exception of f_2. The final results presented in Table 2 show that the predator prey mechanism allowed the PPO to find solutions, on average, orders of magnitude better for f_1, f_2 and f_4. For f_2 the solutions are still significantly better in the statistic sense but the difference is not as meaningful as for the other functions. We can conclude that, in the experimental setup we used, the PPO seems to have three advantages over the best PSO tested: significantly better solutions are found, the algorithm converges faster and, since better solutions are still being found at the end of the runs, the PPO can effectively escape local sub-optima. These results apparently support our claim that the predator prey mechanism can constitute an effective way of avoiding premature convergence to local sub-optima in particle swarm based optimisers.

4 Conclusions and Future Work

We presented a new swarm particle based function optimiser, inspired in the natural relation between predator and prey. A predator-prey mechanism was implemented in

addition to the standard swarm particle optimiser, obtaining a new algorithm that was tested in four benchmark functions against three variants of the standard PSO. The experimental results allow us to conclude that the PPO performed significantly better than the standard PSO in the optimisation of four benchmark multimodal functions, either by finding, on average, better solutions an the end of the runs, or by finding solutions of similar quality but with a higher convergence rate.

As for the main direction of our future work with the predator prey optimiser, it will probably lead to the development a multi-predator sub-population version of the algorithm.

Acknowledgements. This work was partially financed by the Portuguese Ministry of Science and Technology under the Program POSI.

References

1. Kennedy, J. and Eberhart, R. C., "Particle swarm optimisation", Proc. IEEE International Conference on Neural Networks. Piscataway, NJ, pp. 1942-1948, 1995.
2. Kennedy, J., Eberhart, R. C., and Shi, Y., "Swarm intelligence", Morgan Kaufmann Publishers, San Francisco. 2001
3. Angeline, P. J., "Evolutionary optimisation versus particle swarm optimisation: philosophy and performance differences", The Seventh Annual Conf. on Evolutionary Programming. 1998.
4. Shi, Y. and Eberhart, R. C., "Parameter selection in particle swarm optimisation" Evolutionary Programming VII: Proc. EP 98. New York, pp. 591-600, 1998.
5. Shi, Y. and Eberhart, R. C., "Empirical study of particle swarm optimisation", Proceedings of the 1999 Congress on Evolutionary Computation. Piscataway, NJ, pp. 1945-1950, 1999.
6. Clerc, M. and Kennedy, J.. "The particle swarm-explosion, stability, and convergence in a multidimensional complex space". IEEE Transactions on Evolutionary Computation, Vol. 6, No. 1, pp. 58-73. 2002.
7. Kennedy, J., "Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance", Proc. Congress on Evolutionary Computation 1999. Piscataway, NJ, pp. 1931-1938, 1999.
8. Muhlenbein, H., & Schlierkamp-Voosen, D.. "Predictive models for the breeder genetic algorithm: I. Continuous parameter optimisation." Evolutionary Computation, 1 (1), 25-49.
9. Eberhart, R. C. and Shi, Y. "Comparing inertia weigthts and constriction factors in particle swarm optimization". Proc. CEC 2000 pp. 84-88. San Diego, CA, 2000.

Data Mining Support for Case-Based Collaborative Recommendation*

Barry Smyth, David Wilson, and Derry O'Sullivan

Smart Media Institute
Department of Computer Science
University College Dublin
{barry.smyth,david.wilson,dermot.osullivan}@ucd.ie

Abstract. This paper describes ongoing research which aims to enhance collaborative recommendation techniques in the context of PTV, an applied recommender system for the TV listings domain. We have developed a case-based perspective on PTV's collaborative recommendation component, viewing the sparsity problem in collaborative filtering as one of updating and maintaining similarity knowledge for case-based systems. Our approach applies data mining techniques to extract relationships between program items that can be used to address the sparsity/maintenance problem, as well as employing recommendation ranking that combines user similarities and item similarities to deliver more effective recommendation orderings.

1 Introduction

Modern recommender systems typically employ collaborative, content-based methods, or some combination thereof [1,2,3,4,5,6,7]. Content-based techniques compare items by using their feature-based descriptions, choosing to recommend items that are most similar to those a user has previously liked. However, such methods incur a knowledge engineering overhead associated with compiling rich item descriptions. Furthermore, recommendation diversity can be compromised as new recommendations are, by definition, similar to items in the user's profile.

In contrast, collaborative techniques make recommendations by comparing user profiles of rated items, according to similarities and differences in the item ratings. Collaborative techniques require little knowledge-engineering overhead, and there is greater opportunity for recommendation diversity. However, collaborative techniques incur a cost in gathering enough profile information to make accurate user similarity measurements. This *sparsity problem* tells us that, on average, two users are unlikely to have rated many of the same items and so there will be little direct overlap between their profiles. This is problematic when it comes to retrieval, because it means that there may be no direct way to measure the similarity between two profiles unless we have access to additional knowledge that allows us to compare non-identical profile items.

PTV is an established online recommender system deployed in the television listings domain which combines complementary results from separate user-based collaborative

* The support of the Informatics Research Initiative of Enterprise Ireland is gratefully acknowledged.

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 111–118, 2002.

and case-based components for recommending programs to users [4]. In this research, we are interested in improving the recommendations of the overall system by addressing the sparsity problem in the collaborative component. Sparsity can be addressed directly by compelling users to make additional rankings tailored to increase recommendation breadth. Instead of presenting users with this unwelcome burden, we would like to base our efforts on the information afforded through the normal task-based processing of the system. We are mindful, however, that the collaborative filtering component is desirable, in part, because it can provide diverse and high-quality recommendations with minimal knowledge-engineering effort. Data mining techniques have been used in other research [8]; our approach differs with respect to the use of the knowledge generated. We employ the association rules to enhance collaborative filtering algorithms, rather than using the rules directly for rule-based recommendation.

Addressing the sparsity problem faced by collaborative systems is very much in the spirit of our experiences with reasoning knowledge maintenance in case-based reasoning (CBR) [9]. CBR would typically be viewed well within the content-based recommendation camp, but the methodology of retrieving similar user profiles (cases) and adapting them by combining new items with the current profile (case) context lends itself to a case-based view [10]. With this in mind, we have forwarded a case-based perspective on collaborative filtering [11], and we approach the problem of reducing sparsity as a similarity coverage problem in CBR, where similarities are based on an incomplete (possibly non-overlapping) space of descriptors that calls for augmentation (c.f., [9,12]).

This paper describes our data mining approach to improving case-based collaborative recommendation, and it presents experiments that show the benefits of mining similarity knowledge to address sparsity, both at the level of user-based recommendation and at the level of recommendation ranking. From the collaborative perspective, this provides a way to address the sparsity problem. From the CBR perspective, it provides a new approach to similarity maintenance.

2 Mining Program Similarity Knowledge

There are many automated techniques that could be used to derive item similarity knowledge. The initial approach we have chosen is to apply data mining techniques (see [13] for an overview), in particular the *Apriori* algorithm [14], to extract association rules between programmes in PTV user-profile cases. By discovering hidden relationships between TV programmes, we may be able to cover more potential profile matches and, at the same time, make more informed recommendations [11]. For example, under conventional collaborative filtering techniques, a person that likes *X-Files* and *Frasier* would not normally be comparable to a person that likes *Friends* and *ER*, but discovering a relationship between *Frasier* and *Friends* would provide a basis for profile comparison.

2.1 Item Similarities

By treating PTV user profiles as transactions and the rated programmes therein as itemsets, the Apriori algorithm can be used directly to derive a set of programme-programme association rules and confidence levels. We have limited this phase of the work to rules

with single-programme antecedents and consequents. Confidence values are taken as probabilities and used to fill in a programme-programme similarity matrix, which provides the additional similarity knowledge necessary to compare non-identical profile items [11].

Since the matter of additional similarity coverage rests in populating the matrix as densely as possible, a natural extension suggests itself. Directly generated rules can be chained together ($A \Rightarrow B$ and $B \Rightarrow C$ imply $A \Rightarrow C$) to provide indirect program relationships. A choice then has to be made as to how the indirect rule confidence will be calculated (e.g., minimum, maximum, or some combination of the confidences in the potential paths); our experiments present results from a number of different models. Rule similarity knowledge that is generated by Apriori, we refer to as *direct*, and additional derived knowledge as *indirect*.

Building such item-item similarity knowledge to address the sparsity/maintenance problem is similar in spirit to item-based collaborative techniques [15]. The item-based collaborative approach uses rating overlaps to build item-item similarities, and suggested items are then retrieved in direct comparison to the elements that comprise a user profile. The direct nature of such item retrieval recalls direct content-based item recommendation, as well as the potential cost in terms of diversity.

3 Recommendation Strategy

The availability of item similarity knowledge facilitates a new type of similarity-based recommendation strategy that combines elements from case-based and collaborative recommendation techniques. It facilitates the use of more sophisticated CBR-like similarity metrics on ratings-based profile data, which in turn make it possible to leverage indirect similarities between profile cases, and so generate improved recommendation lists. This new recommendation strategy consists of two basic steps:

1. The target profile, t is compared to each profile case, $s_i \epsilon S$, to select the k most similar cases.
2. The items contained within these selected cases (but absent in the target profile) are ranked according to the relevance to the target, and the r most similar items are returned as recommendations

3.1 Profile Matching

The profile similarity metric is presented in Eq. (1) as the weighted-sum of the similarities between the items in the target and source profile cases. In the situation where there is a direct correspondence between an item in the source, c_i, and the target, t_j, then maximal similarity is assumed (Eq. (2)). However, the nature of ratings-based profile cases is such that these direct correspondences are rare and in such situations the similarity value of the source profile item is computed as the mean similarity between this item and the n most similar items in the target profile case ($t_1, ..., t_n$) (Eq. (3)).

$$PSim(t, c, n) = \sum_{c_i \epsilon c} w_i \cdot ISim(t, c_i, n) \tag{1}$$

$$ISim(t, c_i, n) = 1 \ \ if \ \exists \, t_j = c_i \tag{2}$$

$$= \frac{\sum_{j=1..n} sim(t_j, c_i)}{n} \tag{3}$$

Notice, that if $n = 1$ and there is a perfect one-to-one correspondence between the target and source profile cases, then this profile similarity metric is equivalent to the traditional weighted-sum similarity metric used in CBR.

3.2 Recommendation Ranking

Once the k most similar profile cases ($\hat{C}$) to the target have been identified a set of ranked item recommendations can be produced. There are three factors to consider when ranking these recommendations. First, we want to give priority to those items that have a high similarity to the target profile case. Second, items that occur in many of the retrieved profile cases should be preferred to those that occur in few profile cases. Finally, items that are recommended by profiles that are similar to the target should be preferred to items that are recommended by less similar profiles. Accordingly we compute the *relevance* of an item, c_i, from a retrieved profile case, c, with respect to the target profile, t, as shown in Eq. (4); where $C' \subseteq \hat{C}$ is the set of retrieved profile cases that contain c_i.

$$Rel(c_i, t, \hat{C}) = ISim(c_i, t, k) \cdot \frac{|C'|}{|\hat{C}|} \cdot \sum_{c \epsilon C'} PSim(c, t, k) \tag{4}$$

4 Experimental Evaluation

In order to evaluate our approach to mining and applying similarity knowledge, we conducted a series of experiments using data from 622 PTV user profiles. The first set of experiments were designed to investigate the performance characteristics of our chosen data mining algorithm within the PTV domain. The second set of experiments tested the potential for mining additional similarity knowledge, in terms of relationships between program items. The third set of experiments tested the potential of the approach for improving actual recommendation quality. With encouraging results on the PTV data set, we repeated our experiments using a set of 659 user profiles from the MovieLens data set, to test the approach on a different dataset and in a different domain. For experimental runs, we examined only positively rated items, leaving negative ratings for future work.

4.1 Tuning Data Mining

In our first set of experiments, we applied the Apriori algorithm to both datasets for different parameterizations of the algorithm. Since data mining was the basis for maintaining similarity knowledge, we wanted to determine how rule generation would be influenced by parameter choice, namely confidence and support thresholds. Previous experiments demonstrate how we tune the Apriori algorithm to provide optimal results for use in our recommendation techniques. It was found that the number of rules generated (and rule accuracy) is more dependent on confidence than on support [11]. Based on these results,

we chose representative support levels (PTV: 5%, MovieLens: 20%) for the remainder of the experiments. Having different representative support values highlights an important difference; the PTV dataset was much sparser than MovieLens. This caused lower rule generation at high support values, which necessitates dropping to low support values to find more rules with higher confidences.

4.2 Increasing Similarity Coverage

In the next set of experiments, we were interested in evaluating the degree to which similarity coverage could be improved by the rule sets. For the first experiment, the density of the generated item-item similarity matrix was taken as our measure of similarity coverage. We varied confidence levels from 40% to 5% at 5% intervals. On each run we generated the Apriori direct rule set, as well as a maximal indirect rule set and filled in the item similarity matrix, taking the matrix density relative to the items that participated in the rule set. The direct item similarities provide an average of 10% coverage, but there is a marked increase for the indirect similarity rules. We note a maximum of approximately 65% coverage in the best case. Of course, the Apriori method does not generate association rules for every profile program, and in fact Apriori ignores a great many programs because their frequency fails the Apriori threshold tests. Nevertheless when we add these newly generated rules to the full item-item matrix, we found an increase in density from 0.6% to 2.6%. Again, a similar pattern was found in the MovieLens data, .08% to .36%.

4.3 Improving Recommendations

With encouraging results for increasing the similarity coverage, we designed experiments to test the effect of this new similarity knowledge on recommendation quality. Based on our earlier experiments, we chose representative confidence levels of 10% on the PTV dataset and 70% on the MovieLens dataset.

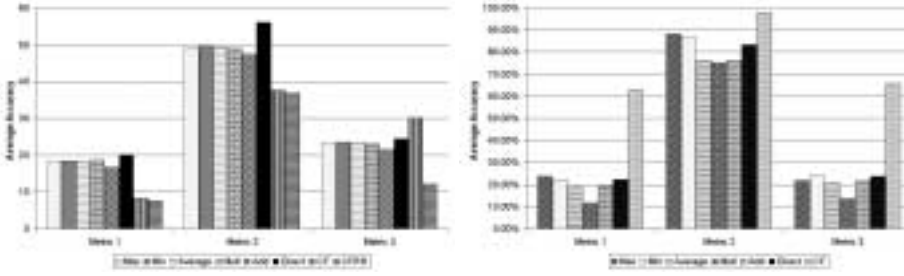
Recommendation Coverage. In the first experiment, we measured the number of profiles in the PTV dataset that could possibly be compared as a percentage of all potential profile comparisons for different similarity metrics. Using our standard collaborative filtering metric gave coverage of 42%, while our similarity metric using direct rules only and then adding indirect rules increased the coverage to 52% and 70% respectively.

Recommendation Accuracy. The final experiment tested the accuracy of recommendation on both datasets. After generating direct and indirect similarity rules, a profile case is selected from the case-base. A parameterized percentage of the items in the selected case are removed from consideration. The remainder of the profile case is then used as the basis for retrieval, using our similarity metric and recommendation ranking. Accuracy was calculated using three metrics:

1. Percentage of removed items that were recommended in each case.
2. Percentage of profiles in which at least one removed item was recommended

3. A combination of the above: percentage of removed items that were recommended, given a recommendation could be made

We were most interested in looking for uplift in recommendation quality when comparing our CBR technique (using direct and indirect similarity rules) to a pure collaborative filtering technique. We tested the effect of our recommendation ranking function by correlating rank score/numerical rank and item success. On the PTV dataset, results of the first metric showed that using direct rules (20%) outperformed indirect rules (18%), which in turn outperformed pure CF (8%). A similar pattern was seen in the other metrics except for metric 3 which showed pure CF having a higher accuracy (30%) than either CBR method (Direct: 24%, Indirect: 30%) (Fig. 1). These results seem to indicate that indirect rules



(a) (b)

Fig. 1. Recommendation accuracies for a), PTV and b), Movielens

may not be as useful as their direct counterparts; another possibility is that these rules are generating high quality recommendations which are not present in our profile cases.

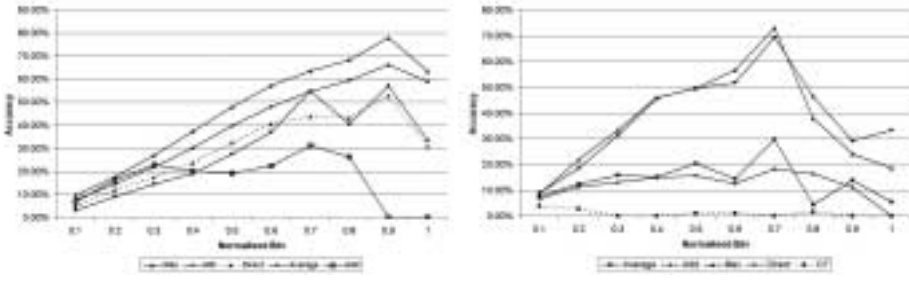
On the MovieLens dataset, CF outperforms both direct and indirect rules in all three metrics (by almost 30% on average) (Fig. 1). These results may indicate that more densely populated profile case-bases favour CF techniques, an issue that bears further study for possible multi-strategy approaches.

Recommendation Ranking Accuracy. In order to test our recommendation ranking function against pure CF methods, we used a simplified version (CFRR) of our recommendation ranking method. For each unique item in our list of recommended profiles c_i, where $C' \subseteq C$ is the set of retrieved profiles that contain c_i, and a target t:

$$Rel(c_i, t) = \frac{|C'|}{|C|} \cdot \sum_{c \epsilon C'} CFSim(c, t) \tag{5}$$

We then test our ranking functions by finding correlations between the score/rank of an item and its success over all profiles. For recommendation score, we normalize the score and then find the number of successes in each bin (0–1.0 in 0.1 increments); for rank, our bins contain all the successes at each rank position over all profiles. Results

show that certain models (Maximum, Addition, Average) used in chaining for indirect rule generation give greater correlations over both direct (14% increase rank, 19% score) and CF techniques (52% increase rank) (Fig. 2). We also found that score binning proved



(a) (b)

Fig. 2. Rank Score accuracies for a), PTV and b), Movielens

extremely inaccurate under CF with little or no correlation to be found. With MovieLens data, we again found that certain models provide uplift from direct to indirect rules (8% rank, 4% score) (Fig. 2). Again CF results proved inaccurate for use. When comparing both datasets ranking correlations, we also note that using the Maximal model for rule chaining provides the best possible accuracy.

5 Conclusion and Future Work

We have described a data mining approach to ameliorating the problem of sparse similarity knowledge in profile-based recommendation systems. From a collaborative filtering standpoint, this provides a means to address the sparsity problem. From a case-based reasoning standpoint, this provides a mechanism for maintaining similarity knowledge. An initial evaluation of the work has been conducted in the domain of TV listing recommendation, using a well known commercial recommender (PTV [4]), and the results are supported by additional experiments in the movie recommendation domain. The results provide good evidence in support of the view that learning similarity knowledge through profile mining makes it possible to improve overall recommendation quality.

Our future work will investigate the effects of these algorithms on further data sets, using multiple antecedents and consequents in data mining, and the inclusion of negative ratings. We would also like to compare data mining to item-based collaborative metrics, both in measures such as Mean Absolute Error, as well as in terms of recommendation diversity.

References

1. Herlocker, J., Konstan, J., Borchers, A., Riedl, J.: An algorithmic framework for performing collaborative filtering. In: Proceedings of the 1999 Conference on Research and Development in Information Retrieval: SIGIR-99. (1999)
2. Konstan, J., Miller, B., Maltz, D., Herlocker, J., Gordon, L., Riedl, J.: Grouplens: Applying collaborative filtering to usenet news. Communications of the ACM **40** (1997) 77–87
3. Shardanand, U., Maes, P.: Social information filtering: Algorithms for automating "word of mouth". In: Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems. (1995) 210–217
4. Smyth, B., Cotter, P.: Personalized electronic programme guides. Artificial Intelligence Magazine **21** (2001)
5. Good, N., Schafer, J.B., Konstan, J.A., Borchers, A., Sarwar, B.M., Herlocker, J.L., Riedl, J.: Combining collaborative filtering with personal agents for better recommendations. In: Proceedings of the 1999 Conference of the American Association of Artifical Intelligence (AAAI-99). (1999) 439–446
6. Soboroff, I., Nicholas, C.: Combining content and collaboration in text filtering. In: Proceedings of the IJCAI-99 Workshop on Machine Learning for Information Filtering. (1999)
7. Balabanović, M., Shoham, Y.: Fab: Content-based, collaborative recommendation. Communications of the ACM **40** (1997) 66–72
8. Lin, C., Alvarez, S., Ruiz, C.: Collaborative recommendation via adaptive association rule mining (2000)
9. Wilson, D.C., Leake, D.B.: Maintaining case-based reasoners: Dimensions and directions. Computational Intelligence **17** (2001)
10. Watson, I.: CBR is a methodology not a technology. Knowledge Based Systems **12** (1999) 303–308
11. O'Sullivan, D., Wilson, D., Smyth, B.: Using collaborative filtering data in case-based recommendation. In: Proceedings of the 15th International FLAIRS Conference. (2002) AAAI, Pensacola Beach, Florida, USA.
12. Fox, S., Leake, D.: Learning to refine indexing by introspective reasoning. In: Proceedings of First International Conference on Case-Based Reasoning, Berlin, Springer Verlag (1995) 431–440
13. Hipp, J., Nakhaeizadeh, U.G.: Mining association rules: Deriving a superior algorithm by analyzing today's approaches. In: Proceedings of the 4th European Symposium on Principles of Data Mining and Knowledge Discovery. (2000)
14. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.I.: Fast discovery of association rules. In Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R., eds.: Advances in Knowledge Discovery and Data Mining. AAAI Press (1996) 307–328
15. Sarwar, B.M., Karypis, G., Konstan, J.A., Riedl, J.: Item-based collaborative filtering recommender algorithms. In: Proceedings of the Tenth International World Wide Web Conference. (2001)

The Feasibility of Machine Learning for Query Answering – An Experiment in Two Domains

Richard F.E. Sutcliffe and Kieran White

Department of Computer Science and Information Systems
University of Limerick, Ireland
{Richard.Sutcliffe,Kieran.White}@ul.ie
http://www.csis.ul.ie/staff/richard.sutcliffe

Abstract. We present an experiment in which an information retrieval system using a forest of decision trees was trained using Utgoff's ITI algorithm on two test collections. The system was then compared with a conventional inverted indexing engine and found to give a superior performance. We argue that the method has the potential to be used in real applications where the task domain is homogeneous.

1 Introduction

The goal of Information Retrieval (IR) is to allow users to gain access to the information they need in order to perform some task. Typically the information is contained within a collection of documents stored electronically and the information need is expressed as a short natural language query. A conventional IR system works by comparing terms (i.e. keywords) from the query with those in the documents and returning those documents which have a high degree of match.

The best-known paradigm for carrying out the match is the Vector Space Model (VSM) [11] in which each term in a document is assigned a weight usually based on its frequency in the document (TF) and the inverted frequency in the document collection as a whole (IDF). By considering each possible term as one dimension in a space, a document can be represented as a point in the space. As the same process can be carried out on a query, retrieval becomes a matter of finding the document points which are closest to the location of the query in the space.

VSM has proved an extremely popular and effective approach to IR for three reasons. Firstly, it is equally applicable to documents irrespective of their contents; Secondly, a reasonable level of performance can usually be expected; Thirdly it can be implemented using Inverted Indexing in such a way that retrieval time is not adversely affected by the size of the document collection.

On the other hand, VSM has a number of important disadvantages. While it is effective on heterogenous collections where the topic of documents varies widely, it is much less applicable to homogeneous collections where all documents are about different aspects of the same topic. The reason for this is that

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 119–126, 2002.

documents are being distinguished solely on the basis of the terms they contain. In a homogeneous collection the same terms may well appear in documents whether or not they are relevant because it is the precise context of use which is important. Also, while a system based on VSM will tend to return some correct answers to the average query it is rare for it to perform with very high precision and high recall simultaneously.

As a result of the above points, there has been an increasing interest in the use of Machine Learning (ML) algorithms within IR. Very generally, a supervised ML algorithm is one which can be trained to undertake a task on some pre-defined data for which the correct outputs are known. Following training, the same task can be performed on unseen data leading to the production of substantially correct outputs. ML algorithms are interesting because they can generalise their experiences during training and hence perform appropriately on inputs which do not correspond exactly to any which they have seen before.

Traditionally, experiments in IR involve a document collection and a set of test queries. Each query is associated with a list of the documents previously judged by human experts to contain relevant information. A number of collections are readily available for research purposes. This suggests that an IR system using ML could be trained on some subset of a query collection and then tested on the remainder.

The key objective of the work performed here is to investigate the applicability of an ML algorithm to IR in two domains. The algorithm we use is Incremental Tree Induction (ITI) [14] which is closely related to Quinlan's ID3 [9]. In ID3 there is a set of inputs termed non-categorial attributes each of which can have one of a predetermined number of discrete values. There is one output termed the categorial attribute which may also take one of a set of discrete values. During training a series of ¡input, output¿ pairs is presented to the system which learns to generate an output given its corresponding input. During performance, unseen inputs are presented to the system and the output produced is compared to the correct result in order to evaluate performance of the system. ITI is similar to ID3 but improves on it in a number of ways. For example, ITI allows continuous values to be used for non-categorial attributes. Its main feature however is that it allows for training to take place incrementally between periods of performance. Eventually we envisage employing this ability to implement a relevance feedback mechanism which could refine our system's behaviour over time.

Both ITI and ID3 work by constructing a binary decision tree. At each node a boolean test is carried out on the value of one non-categorial attribute. If the result is true, the left branch is taken while if it is false, the right branch is followed. Each leaf node in the tree has associated with it the value of the categorial attribute which decides the system's output.

Decision trees have been used within information retrieval for a number of purposes which we will review in the next section. In our approach, there is one tree for each document in the collection and one non-categorial attribute for each term occurring in a query. During training, each tree learns to categorise an input

query as either relevant or not relevant to a particular document. As each tree is separate from those corresponding to other documents in the collection and is trained independently, we call the key data structure underlying our approach a Decision Tree Forest (DTF).

We have carried out a number of experiments aimed at investigating the efficacy of the DTF approach. In each case we have compared the performance of DTF with a conventional Inverted Index System (IIS). Tests have been carried out in two application domains, one technical and one scientific. The results show that DTF is superior to IIS in both cases.

The rest of the article is structured as follows. We first review some of the literature on ML and IR focusing particularly on work involving decision trees. We then describe the architecture of our two systems and the characteristics of the training data used in our experiments. Following this we turn to the experiments we have conducted and the results obtained. Finally we draw conclusions from our study and discuss possible next steps.

2 Previous Work on ML and IR

Neural networks have been the subject of IR research on a number of occasions. Possibly the earliest and most well-known example is Belew's work on the AIR project [3]. He developed a three-layer network, relying on relevance feedback to refine the performance of the retrieval system over a period of time. One notable aspect of his research concerns the development of a correlational learning rule to increase the number of connections between documents and index terms.

Genetic algorithms have also been subject to investigation by IR researchers. Gordon [6] created a population of competing document descriptions for the purpose of indexing. Each gene in an individual represented a document keyword. Over successive generations much improved document representatives evolved.

A comparative evaluation by Lewis [7] of a decision tree algorithm, DT-min10, with a Bayesian based system for text categorisation tasks was revealing. His model operated in a manner similar to DTF, using multiple trees. However, whereas Lewis had a tree corresponding to each category we have one for each document. Additionally, the inputs to the DT-min10 trees were documents, but DTF, being a retrieval system takes queries as inputs. An analysis of his results indicated that decision trees handle large numbers of non-categorial attributes better than Bayesian models, but only when a large training set is available. An attribute filtering step was found beneficial for smaller sets.

Decision tree based induction algorithms have already been investigated by other researchers. For example Baudin et al. used ID3 to improve the precision of a conceptual retrieval system [2]. Normally these types of system use heuristics to improve recall, to the detriment of precision. Their approach was to train ID3, based on relevance feedback from users, to filter out heuristics which were ineffective in a particular context.

A probabilistic version of ID3 was used in AIR/X's Darmstadt indexing approach [5]. This is a multi-stage technique, the output of which is a set of

weighted document descriptors representing each document. At each stage an initial set of document descriptors was produced for the individual documents, and the ID3 algorithm estimated the probability that each set of document descriptors was appropriate. In the final stage, ID3's outputs were the document descriptors' associated weights.

Chen et al. [4] employed ID5R (ITI's ancestor) in their inductive approach to query-by-example for IR and database management systems. At query time an induction tree was constructed and refined based on successive relevance feedback iterations.

In the work presented in this paper we use an architecture in which there is one decision tree for each document in the collection. During training this tree learns to recognise the circumstances under which its corresponding document is relevant to a query. By contrast, Chen used one decision tree per query, this being trained interactively at query time. Our approach also differs from the work of Baudin and the Darmstadt indexing of AIR/X in that retrieval is based on decision trees alone whereas these authors incorporated them into a more sophisticated system involving conventional inverted indexing.

3 Architecture of IR Systems

3.1 Decision Tree Forest System

As stated earlier, DTF involves the use of a separate decision tree for each document in the collection being used. Non-categorial attributes (inputs) are binary and there is one for each term occurring in the query set. Each tree in the forest has the same non-categorial attributes. The categorial attribute (output) for a tree is also binary.

For an experiment, the set of queries is divided into a training collection and a test collection. Each query has associated with it a list of relevant documents. During the training of a particular tree, each query in the training collection is converted into an input pattern for the ITI algorithm. This involves tokenising it into words, removing any stopwords that appear and stemming the remainder using the Porter algorithm [8], resulting in that query's terms. For each of these, the value of the corresponding non-categorial attribute is set to True. All other non-categorial attributes are set to False. If the document corresponding to the tree being trained is one of those listed as being relevant to the query, the categorial attribute is set to True. Otherwise it is set to False. The resulting ¡input, output¿ pairs are then used to train the tree. The same is done for all other trees in the forest using the same set of training queries.

During performance, a test query is converted into a decision tree input by setting the appropriate non-categorial attributes and is then presented simultaneously to all trees. If a tree returns True in its categorial attribute, then the document to which it corresponds is deemed relevant to the query. Otherwise it is considered irrelevant. The output for each query is therefore an unranked list of those documents considered relevant.

3.2 Inverted Index System

To provide a means of comparison, we also used an Inverted Index System (IIS) in our experiments. This was developed as part of a previous project and uses the probabilistic matching function described in [10]. During indexing each document in the collection is tokenised, stopwords are removed and the remaining words are stemmed, resulting in a set of document terms. The frequencies with which these occur are recorded in the inverted index. All documents being used in an experiment in either the training or test collections for DTF are indexed in IIS.

During performance, a query in the test collection is tokenised and undergoes stopword removal and stemming. The resulting terms are assigned weights as dictated by the matching function, using the frequencies recorded in the inverted index. These weights vary depending on the document being examined and their sum decides its ranking in the final list of results presented to the user.

4 Training Data

4.1 SIFT Collection

The SIFT collection is based on the instruction manual for the Lotus word processor Ami Pro [1]. The manual describes how to use the system and is divided into 704 numbered sections each of which is considered as a separate 'document' for our experiments. The test collection comprises four sets of queries, each derived from a different source. The Designer and Hyland queries were created by experts. The Schmidt queries were produced by a beginner learning to use the system. Finally, the Orion queries were collected automatically from an e-mail support service. To determine the correct answers to a query, five respondents inspected it independently and decided which sections in the manual were relevant. This data was then combined [13].

For the experiments described here, 229 Orion queries were used. These are characterised as short and highly elliptical. Orion was chosen because it is the only collection containing genuine, naturally occurring queries composed by real users not participating in an experiment.

4.2 Cystic Fibrosis Collection

For our second collection we used the well-known Cystic Fibrosis (CF) queries [12]. There are 1239 documents together with 100 queries each with a list of relevant documents. Again the queries are short, usually only one sentence long.

5 Retrieval Experiments

5.1 Method

In the first experiment, DTF and IIS were compared using the SIFT collection. The queries were divided randomly into 80% for training and 20% for testing.

Section: CHAP17.42.TXT

WordPerfect graphic?
Bring picture or graphic into AmiPro?
How many separate drawings can I import into a frame?
Drawing not working?
Paste a chart from Lotus 1-2-3 to amipro document?

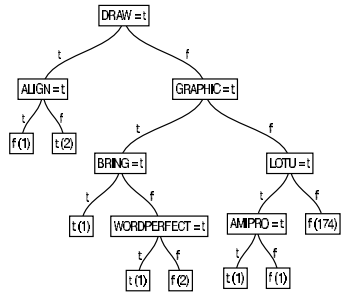


Fig. 1. A section's queries from the SIFT Collection and the resulting tree

DTF was trained using the training queries. One section's queries and the tree constructed from them are shown in Fig. 1.

In order to perform our experiments, two additional ideas were used. Firstly, the Density Value of a document is the number of queries to which it is considered relevant. In training terms, the higher the Density Value, the more likely DTF learning is to work effectively, because more chances are available for the algorithm to learn about the content of the document through inspecting queries about it. Secondly, the Density Threshold is a value which the Density Value of all documents must equal or exceed if they are to be used in an experiment. Three Density Threshold values were employed, 1, 3 and 10. The two higher Density Thresholds were used to simulate an environment where more training data was available to us than there was in reality. To ensure a fair comparison between the two systems, IIS was only indexed on those documents in the collection whose Density Value was greater than or equal to that of the current Threshold.

Because the results returned by DTF are unranked, it was necessary to devise a means of comparing them with the ranked IIS output. This was done as follows. Each test query was presented to the two systems in turn. First DTF returned a set of documents and any which did not reach the Density Threshold were deleted from the list. We will call the number of remaining documents n. Precision, Recall and van Rijsbergen's E-measure [15] were computed on these. Next, the first n documents returned by IIS were used to compute the Precision, Recall and E-measure for this system.

The experiment was conducted four times with different randomisations and the average of the results was then taken. This was to avoid the possibility that a particular randomisation was felicitous for the training algorithm.

In the second group of experiments, the evaluation was repeated by training, indexing and testing on the Cystic Fibrosis Collection, again relying on four different randomisations.

Table 1. DTF and IIS compared using the SIFT and Cystic Fibrosis Collections

Metric	System	Density Threshold					
		SIFT			Cystic Fibrosis		
		1	3	10	1	3	10
Precision	DTF	0.44	0.51	0.64	0.28	0.31	0.45
	IIS	0.49	0.51	0.53	0.39	0.40	0.33
Recall	DTF	0.33	0.38	0.52	0.16	0.19	0.33
	IIS	0.36	0.38	0.38	0.25	0.25	0.25
E Measure	DTF	0.69	0.65	0.59	0.85	0.83	0.70
	IIS	0.67	0.65	0.69	0.77	0.77	0.78

5.2 Results

The results of all experiments are shown in Table 1. In Experiment 1 (SIFT Collection) DTF outperformed IIS at a Density Threshold of ten when measured by all three metrics, and especially so with recall. As the density threshold increases so does the performance of DTF, suggesting that a training collection with high average Density Value is likely to produce a better result with DTF than one with low average Density Value. In Experiment 2 (CF Collection) both systems perform worse than with the SIFT Collection but DTF overtakes IIS at the highest density threshold.

6 Conclusions

The key finding of this study was that DTF surpassed IIS on the two test collections used when the Density Threshold was high. This suggests that DTF is an approach worthy of further investigation. In heterogenous task domains where each query is on a different topic and the same subject rarely comes up repeatedly it is unlikely that a DTF approach would be effective because the system would never be able to learn. On the other hand, in a homogeneous domain such as that of technical manuals, the method has far more promise because the same queries are likely to come up repeatedly and in addition the same domain specific concepts will appear frequently, each time in different combinations. These factors will allow DTF to learn how the meaning of each concept varies depending on the other terms with which it co-occurs in a query.

It seems likely that with an extremely large query set in a homogeneous domain that DTF could learn very effectively and achieve high precision and recall in testing. However, currently such query collections do not exist. Most available ones contain only a few hundred queries. In order to construct a practical system, techniques must be found which increase the potential for generalisation when using a relatively small training set. There are two possible ways in which this might be done. First, query expansion could be undertaken before training by using synonyms or hypernyms. We have observed for example, that many Orion queries mention specific instances of a concept such as ‘deskjet 500’which are

never subsequently mentioned in a query. For this reason, DTF can never learn what a 'deskjet 500' is. On the other hand, if a hypernym of the concept (e.g. 'printer') was added to the query during training then a subsequent query which mentioned 'printer' or a particular type of printer but not specifically 'deskjet 500' could still potentially be deemed answered in the document in question.

A second approach to increasing generalisation might be to use information derived from the document itself rather than just working with queries and treating the document as a black box. Aspects of the query could be associated in some way with terms in the document.

References

1. Ami Pro: Lotus Ami Pro Word Processor for Windows User's Guide Release 3. Atlanta, GA: Lotus Development Corporation, Word Processing Division, 1993.
2. Baudin, C., Pell, B., & Kedar, S.: Using induction to refine information retrieval strategies. In Proceedings of the twelfth national conference on artificial intelligence, pp. 553–559 Seattle, WA, 1994.
3. Belew, R. K.: Adaptive information retrieval, Proceedings of the Twelfth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval, pp. 11–20, Cambridge, MA, 1989.
4. Chen, H. & She L.: Inductive query by examples (IQBE): A machine learning approach. In Proceeding of the 27th Annual Hawaii International Conference on System Sciences (HICSS-27), Information Sharing and Knowledge Discovery Track, Maui, HI, January 4–7, 1994.
5. Fuhr, N., Hartmann, S., Lustig, G., Konstadinos, T., Knorz, G. & Schwantner, M.: Automatic Indexing in Operation: The Rule-Based System AIR/X for Large Subject Fields, 1993.
6. Gordon M.: Probabilistic and Genetic Algorithms for Document Retrieval, Communications of the ACM, 31(10), 1208–1218, 1988.
7. Lewis, D. D. & Ringuette, M.: A comparison of two learning algorithms for text categorization. Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval, pp. 81–93. ISRI; Univ. of Nevada, Las Vegas, 1994.
8. Porter, M. F.: An Algorithm for Suffix Stripping. Program, 14(3), 130–37, 1980.
9. Quinlan, J. R.: Inductive learning of decision trees, Machine Learning, 1986, 1:81-106.
10. Robertson, S. E. & Sparck Jones K.: Simple, proven approaches to text retrieval, Technical Report TR356, Computer Laboratory, University of Cambridge, UK, May 1997.
11. Salton, G. (Ed.): The SMART Retrieval System – Experiments in Automatic Document Processing. Prentice-Hall, Englewood Cliffs, NJ, 1971.
12. Shaw, W. M., Wood, J. B., Wood, R. E. & Tibbo, H. R.: The Cystic Fibrosis Database: Content and Research Opportunities. LISR 13, pp. 347–366, 1991.
13. Sutcliffe, R. F. E. & Kurohashi, S. A.: Parallel English-Japanese Query Collection for the Evaluation of On-Line Help Systems. Proceedings of the Second International Conference on Language Resources and Evaluation, Athens, Greece, 31 May–2 June, 2000, 1665–1670.
14. Utgoff, P. E., Berkman, N. C. & Clouse, J. A.: Decision Tree Induction Based on Efficient Tree Restructuring. Machine Learning journal, 10, pp. 5–44, 1997.
15. van Rijsbergen, C. J.: Information Retrieval. London: Butterworths, 1979.

Meta-knowledge Annotation for Efficient Natural-Language Question-Answering

Tony Veale

Computer Science Dept., University College Dublin
Tony.veale@ucd.ie

Abstract: A recent trend in the exploitation of unstructured text content is the use of natural language question answering (NLQA) systems. NLQA is an elaboration of traditional information retrieval techniques for satisfying a user's information needs, where the goal is not simply to retrieve relevant documents but to additionally extract specific passages and semantic entities from these documents as candidate answers to a natural language question. NLQA is thus a tight integration of natural language processing (NLP), information retrieval (IR) and information extraction (IE) designed to circumvent the deep and brittle analysis of questions in favor of shallow but robust comprehension, to ultimately achieve a broad domain question-answering competence. It is argued here that the key to achieving good quality answers in a high-throughput setting lies in a system's ability to construct rich queries that incorporate knowledge from multiple sources.

1 Question Answering Systems

Question-answering systems that exploit information-retrieval are predicated on two major assumptions:

1. Most interesting questions have already been answered somewhere, so NLQA can be conceived as a shallow textual search for existing answers rather than a deep logical proof of novel ones (see [1], [5]).
2. The textual form of a question tends to resemble the textual form of its answer. Thus an IR query constructed from a question text will be effective in retrieving documents that contain answers.

Though neither of these assumptions is defensible on theoretical grounds, there are strong empirical grounds for adopting them both, insofar as they yield computationally tractable systems that offer satisfactory service to users. These assumptions shape the general approach to IR-based NLQA as follows:

1. Given a user question *Q*, generate a natural-language representation *nlp(Q)*
2. From *nlp(Q)*, generate an information-retrieval query *ir(nlp(Q))*
3. Use *ir(nlp(Q))* to retrieve *D* documents from an authoritative text archive
4. If *D* is too small or too large, modify *ir(nlp(Q))* accordingly and return to 3

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 127-134, 2002.

5. Analyze the contents of *D* to derive *A* candidate answer passages from *D*
6. Perform semantic labeling to extract the set of named-entities *E* in *A*
7. If *E* is too small or not compatible with *nlp(Q)*, modify *ir(nlp(Q))* accordingly and return to 3
8. Rank and display <*A, E*> according to the semantic target of *Q* (i.e., who, when, where, etc.)

NLQA systems primarily differ as to the complexity of *ir(nlp(Q))* and the ideal sizes for *D* and *A*, the number of retrieved documents and candidate answer passages. When *ir(nlp(Q))* is richly structured and highly constrained, *D* is typically small and the work load of the IE component is low. But when *ir(nlp(Q))* is simple (a bag of conjoined words, say), then *D* is typically large and the IE component can become a bottleneck, particularly in a commercial setting where question throughput is almost as important as answer precision. To reduce recall to a manageable level, a system can make *ir(nlp(Q))* as constrained as possible. However, this can have the opposite effect of staunching recall completely, so that the system is unable to provide any answers at all. To strike an effective balance, an NLQA system employs an iterative search to incrementally modify *ir(nlp(Q))* until sufficient documents can be found to supply a convincing answer. NLQA is thus a heuristic search process through the space of possible *ir(nlp(Q))* formulations, until one is found to semantically fit the available text data. The unstated goal of commercial NLQA is to guide this tour of the search-space in as efficient a manner as possible without sacrificing competence.

The formulation of *nlp(Q)* has been discussed in depth elsewhere [3]; it suffices to state here that it presupposes a shallow NLP analysis in which multiword phrase boundaries are recognized, parts-of-speech are assigned to each word or phrase, head-words are identified and named entities (such as people and companies) are appropriately tagged. Instead, this paper focuses on a means of formulating *ir(nlp(Q))* that provides as much knowledge as possible to the heuristic processes that will maximize NLQA throughput, by intelligently directing the modification and exploitation of *ir(nlp(Q))*.

2 Annotations and Strategies

The iterative nature of NLQA suggests that it is a non-deterministic integration of NL, IR and IE: when IE is unsatisfactory, IR must be regenerated which involves a reconsideration of NL. This non-determinism requires that each component maintain an internal state so that it may be reactivated as the search process re-iterates. This not only increases and complicates the mutual dependence between components, it introduces considerable overhead from a session-tracking perspective into a high-volume NLQA system. To make these components mutually independent and largely state-free, *ir(nlp(Q))* is thus conceived as a one-off meta-query, whose annotations contain knowledge to drive future modification and evaluation processes without the involvement of the NL and IR components. Since these annotations mark the relative importance of each part of the query (as determined by NL analysis), a richer scoring mechanism can be used to evaluate retrieved answers. This approach has broad impli-

cations, since it transforms *ir(nlp(Q))* from a simple by-product of knowledge to an active knowledge component in its own right.

Annotations primarily allow a query to be iteratively modified in two ways:

1. A non-focal term can be *pruned* if it is does more to reduce recall than increase precision.
2. A term can be *expanded* by adding synonyms and other correlated disjuncts.

The purpose of both strategies is to increase the recall of a tightly constrained query. We do not consider strategies that also reduce recall, since this leads to a combinatorial search space of possible modification sequences. Thus we assume that *ir(nlp(Q))* as initially formulated is maximally constrained, so that it contains each non-stopword of the question, it uses proximity operators to tie modifier terms to their heads, and uses phrasal operators to preserve the internal ordering of named entities.

As for meta-knowledge, the pruning strategy requires that a query be annotated as to the answer utility of each keyword or query fragment. Utility is discussed in the next section, so for now it suffices to note that certain elements of a question are more focal to the user's information need than others, and that the most focal elements should be the last to be pruned from a query. The expansion strategy requires that all possible disjunctions (synonyms, holonyms, partonyms, etc.) are already present in the query as annotations, so that expansion is an in-place conversion of an annotation to a query operator. We do not consider here strategies that weaken the proximity constraints in distance-based operators, assuming instead for simplicity that these constraints have sufficient breathing space in their original specification.

The query language used is that of Inquery, which uses a prefix notation not unlike that of Prolog [2]. The conjunction operator is *#and*, the disjunction operator is *#or*, the phrase or contiguity operator is $\#od_n$ (where n >= 1 is the allowable gap between words), and the proximity or window operator is $\#uw_n$ (where n is the maximum window size). Additionally, the *#syn* operator is used to indicate synonymy at the word and phrase level. To this basic set we add following meta-operators, which can be used to insert annotations at any level of a query:

1. *#meta_alpha*	Marks a keyword or fragment as having high answer utility
2. *#meta_beta*	Marks a keyword or fragment as having indifferent utility
3. *#meta_gamma*	Marks a keyword or fragment as having low answer utility
4. *#meta_syn*	Specifies possible disjuncts for the first argument term
5. *#meta_utility*	Associates a numeric utility with a term or fragment
6. *#meta_target*	Specifies a semantic type that must be present in an answer

When a meta-query is used to retrieve documents, the annotations are recursively removed by discarding all but the first argument of every meta-operator. Since meta-operators are not idempotent, a term annotated by two nested *#meta_alpha* operators is attributed a much higher answer-utility than a term annotated with just one. Likewise, a term annotated with two *#meta_gamma* operators possesses little utility.

The expansion strategy works by simply converting a *#meta_syn* operator into a *#syn* operator. As a result, each annotation argument is transformed from meta-data

into query-data, by becoming a *#syn* argument that is actually used for document retrieval. Therefore, expansion only applies to *#meta_syn* nodes that have literal query fragments or terms as arguments. Pruning is also straightforward to perform, and is not as drastic a strategy as it might first seem. This is because a question essentially contains two kinds of content words: those that are best used to retrieve candidate answers, and those that are best used to evaluate the semantic fit of those answers to the question. While these kinds are not mutually exclusive, some kinds of words are clearly suited to one purpose rather than another. For instance, nouns are generally best suited for retrieval since they often determine the domain of discourse, while adverbs are best suited to evaluation since they tend to reflect a subjective opinion about a situation. So when a term is pruned from a query, it is not actually discarded but merely moved, from the retrieval set to the evaluation set. Indeed, since evaluation occurs post-retrieval, greater computational effort can at this point be spent fitting an evaluation term to an answer (e.g., using WordNet-based similarity measures [4]).

3 Answer Utility

Pruning is thus used to ensure that those terms that most reduce recall while least improving precision are placed in a question's evaluation set rather than its retrieval set, despite their linguistic class. For example, some terms are used to facilitate the framing of the question rather than to shape its semantic content, as in the use of the word "think" in "What does Darwin think about acquired characteristics". But to recognize these framing terms for what they are – contextual noise – requires a powerful grammar and rich model of language use. Alternately then, we do not exclude them from the query but contrive to ensure that such terms are amongst the first to be pruned. This contrivance takes the form of *answer utility*, a measure that attempts to quantify the harm that would be done to answer quality if a retrieval term were to be pruned.

Several factors combine to yield the answer utility of a term. Syntactic utility is an estimate of utility based on the syntactic function of a term, while semantic utility estimates the contribution of a term to the meaning of the question and thus, indirectly, to the meaning of the answer. Named entities such as people names, companies and products have the highest syntactic utility, since they almost always establish the focus of the user's information need. So for example, in the question "Who is the CEO of Vantive", the term "Vantive" has higher syntactic utility than "CEO", which is to say that we should prefer to prune the latter before the former. We thus consider named entities to be *double-alpha* terms, which means that these terms are annotated with two nested *#meta_alpha* operators in a query. Of slightly less utility are nouns, which are the alpha-terms of a question since they serve to anchor it in a particular domain, while verbs and adjectives are viewed indifferently as beta-terms. Finally, adverbs as treated the gamma-terms, since the meaning carried by an adverb in a question may be expressed in an answer via an adjective, a verb or a noun; this mutability makes for poor retrieval so this is reflected in a lower perceived utility.

Semantic utility can be estimated when an NLQA system has direct access to the indices of the underlying IR engine, so that term co-occurrence probabilities and estimated mutual information measures can be used to determine the discrimination power of a term with respect to the question as a whole. Though space precludes a full

discussion here, the general method is briefly outlined. First, it is necessary to capture the semantic focus of the question. Using conditional term probabilities, the focus can be modeled as the term that best predicts the question as a whole. (e.g., in "Who was the captain of the Titanic", the focus is Titanic, not captain). Next, the focality of every other term in the question is estimated, by calculating the mutual information between each term and the focus. The redundancy of each term is then estimated, again using conditional term probabilities, as the extent to which the question as a whole predicts the given term (redundancy is thus the converse of focality). Finally, the semantic utility of each term is estimated as a function of focality and redundancy; there are many ways this function can be formulated, but in general, utility will be proportional to focality and inversely proportional to redundancy.

The operator *#meta_utility* is used to annotate the leaf terms of a query with its corresponding measure of semantic utility. Higher-level annotations will additionally capture the syntactic utility of these terms, so that a recursive measure of utility can be recursively calculated for any fragment of the query. This allows the fragment of weakest utility to be determined at any point and pruned (but note: the leaf terms of the pruned fragment are placed in the evaluation set for any retrieved answers).

4 Cost-Driven Search

NLQA is a hill-climbing search through the space of possible text answers to a question. The start state of this search is determined by the query *ir(nlp(Q))*, which is initially constructed to be maximally constrained and gradually weakened via expansion and pruning. Though NLQA is fundamentally an IR process, it wields a key advantage over conventional IR, since a natural language question specifies a semantic target (e.g. PERSON for "who" questions, PLACE for "where" questions) that serves as valuable meta-knowledge for the search. It allows a system to determine whether the goal state has truly been found, or whether more query modification is required even when apparently relevant documents have already been found. Thus, if an entity of the desired type cannot be extracted from an otherwise relevant document set, NLQAs system will prolong a search that an IR process would instead terminate.

Pruning and expansion are the state-traversal operators that allow NLQA to navigate the space of possible answers. Each strategy is assumed to select and modify a single node of the query at a time, but because pruning is destructive, the order of modifications needs to be explicitly scheduled. A governing principle of *least cost modification* is needed to determine the order of this schedule, where this cost reflects the loss of answer utility inflicted on the query by a modification. For pruning, the modification cost is equivalent to the answer utility of the term to be pruned, so the least useful terms are always pruned first. For expansion, the cost is an estimate of the decline in term utility when less reliable disjuncts are introduced (as discussed next). At each stage of the search, each term is considered both for pruning and for expansion (if appropriately annotated with additional disjuncts). Simply, the term/strategy combination that yields the least non-zero cost is selected, so the precision of the remaining query is least affected.

5 Sources of Disjunction

As there are no true synonyms, the introduction of disjunctive elements into a query must necessarily reduce its precision. The decline is affected by a number of factors, such as the intrinsic ambiguity of a term (i.e., the number of different WordNet synsets it appears in [4]) and more importantly, the intrinsic ambiguity of each synonym. For example, a query can be seriously diminished by the introduction of a highly polysemous term into a query as a disjunctive alternative for a relatively homonymous question term (e.g., introducing "bug" as a synonym for "glitch"). The cost of expansion should thus reflect the potential decline in precision, and as such, should be a direct function of both the answer utility of the term being expanded, and of the total polysemy of the expansion set being introduced. The former quantifies the utility that can be lost in the expansion, while the latter essentially quantifies the number of ways in which it might be lost.

Recall that the enabling assumption of IR-based NLQA is that questions significantly resemble their answers. But the introduction of too many disjunctive forms into a query can result in answers that look nothing like the question. Furthermore, if such an answer also proves to be of low relevance, it may not be apparent to the user why it was generated, and users will simply not use systems they cannot trust. Expansion should thus be minimized, by introducing disjunction into a query in a gradual fashion so that when possible, answers can be retrieved using the original text of the question. Now, because all the possible disjuncts of a term are already present in a meta-query as annotations, the rate of expansion is completely governed by the structure of the query. A gradual deployment of synonym knowledge can be achieved by organizing the disjunctive set of a term into layers, so that the strongest synonyms are expanded first while more tenuous members require successive expansions to be revealed.

The disjunctive set of a term is constructed from a variety of conceptual relations, as derived from lexical knowledge sources like WordNet, a good source of synonyms, hypernyms and hyponyms. Statistical techniques can be also used to extract significant other terms from the English glosses that accompany each WordNet definition (e.g., the gloss of "surgeon" yields "surgery"). Yet perhaps the most useful knowledge source is the finite set of irregular verb forms, since the most common verbs are irregular (e.g., knowing that "spun" is a disjunct of "spin", or vice versa, can be invaluable). These sources are considered in order of similarity when constructing the annotation structure of a query, so that the disjuncts least likely to diminish precision (such as irregular forms) will be considered for expansion first. The intent of course is to ensure that the least important terms are expanded before more focal ones, and that any expansions are done with the most reliable disjuncts first. Consider for instance the following ordering:

- β: *Past, Past-of, Plural, Plural-of, Past-participle, Past-part-of, Nominal, etc.*
- γ: *Synonym, Hyponym*
- $\gamma\gamma$: *Hypernym, Partonym, Holonym*

The schedule is organized according to the utility of each relation type: the irregular forms are viewed as beta relations that introduce no loss of precision into a query, and are thus rewarded with a *#meta_beta* annotation that reflects their indifferent

syntactic utility. In contrast, synonyms and hyponyms are viewed as potentially dangerous gamma-terms, while the lowest double-gamma rating is reserved for hypernyms, partonyms and holonyms. Note that the alpha annotation is not used given to any relation type on the schedule, since this is reserved for the original term as it appears in the question. This means that answers containing actual question terms are more valued than those that simply contain related disjuncts.

Figure 1 illustrates the annotated query structure generated from the question "Who was the *captain* of the *Titanic* when she was *struck* by an *iceberg* and *sank*".

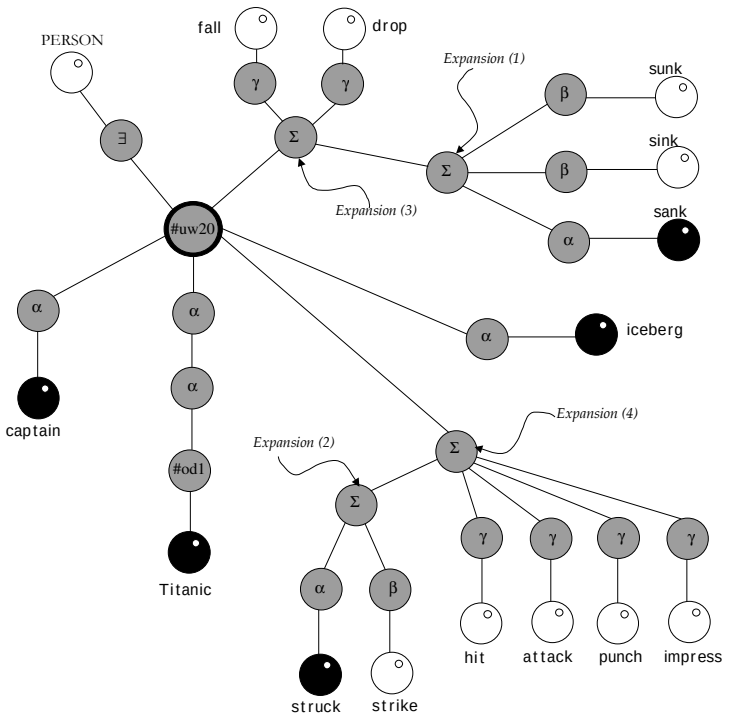


Fig. 1. The annotated structure of a meta-query. Annotations are shown using Greek letters, with Σ denoting *#meta_syn* and ∃ denoting *#meta_target*. The sequence of expansions is marked. White nodes depict annotation data, not query arguments, until an expansion occurs

6 Evaluation and Concluding Remarks

Meta-knowledge annotations are used to maximize question throughput without diminishing answer quality. The approach was therefore evaluated in a performance comparison between CoreAnswer, a product of Coreintellect inc. that uses rich query annotations, and Lasso, the top-rated system in the 1999 TREC QA evaluation (in the 250-byte answer category, [5]). Lasso (licensed by Coreintellect under the name Arrow in 1999) uses the same IR framework for NLQA but employs a flat query formulation, compensating for this lack of structure by its willingness to retrieve and evalu-

ate many hundreds of documents per question. Lasso achieved good answer quality on the Coreintellect archive of authoritative business content (around 5 million articles), as measured against a benchmark set of 100 business questions such as "How much did Peoplesoft pay for Vantive" (note that Lasso is being steadily improved, using techniques such as abductive text inference [6]). However, Lasso's large retrieval sets created considerable throughput problems, with many questions requiring over 30 seconds of CPU time. CoreAnswer was designed as an alternative that would maintain quality but drastically improve throughput.

On the same set of business questions, CoreAnswer produced answers that were equal or better in quality that Lasso for 92% of the questions. More significantly, CoreAnswer retrieved and analyzed an average of 23 documents per question, compared with an average of 617 documents for Lasso. Overall, CoreAnswer required an average of 3.7 seconds of CPU time per question, compared with 27.2 seconds per question for Lasso on the same platform. This increase in performance is clearly due to the dramatic decrease in retrieval size, with only a concomitant drop in quality of 8%. As such, we consider the meta-query approach to be a promising one, though considerable empirical analysis is still required to determine the full extent of the improvements that it offers.

References

1. Burke, R., Hammond, K., Kulykin, V., Lytinen, S., Tomuro, N., Schoenberg, S. (1997). Natural Language Processing in the FAQ Finder System: Results and Prospects. *Working Notes of the Spring AAAI Symposium on the World Web Web*, 1997.
2. Callan, J. P., Croft, W.B., Harding, S.M. (1992). The INQUERY Retrieval System. *In the proc. of the 3rd International Conference on Database and Expert Systems Applications*. 78–83.
3. Kwok, C., Etzioni, O., Weld, D. W. (2001). Scaling Question Answering to the Web. *ACM Transactions on Information Systems*, 19(3). 242-262.
4. Miller, A. G. (1995). WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11).
5. Moldovan, D., Harabagiu, S., Pasça, M., Mihalcea, R., Goodrum, R., Girju, R., Rus, V. (1999). Lasso: a tool for surfing the answer net. *In the Processings of TREC-8*, 1999.
6. S. Harabagiu & S. Maiorano (1999). Finding Answers in large collections of texts: paragraph indexing + abductive inference. *Working Notes of the Fall AAAI Symposium on Question Answering*, Nov.1999.

Financial Time Series Modelling Using Neural Networks: An Assessment of the Utility of a Stacking Methodology

Anthony Brabazon

Dept. of Accountancy, University College Dublin, Ireland
Anthony.Brabazon@ucd.ie

Abstract. This study examines the ability of a series of neural networks (MLPs) to predict the five day percentage change in the value of the FTSE 100 Index, during the period June 1995 to December 1996, using technical, fundamental and intermarket data. The primary findings are consistent with a hypothesis that neural network models are capable of detecting structure in the underlying data but indicate that predictive accuracy declines as the time lapse from the model building period increases.

1 Introduction

The objective of this study is to determine whether a multi-layer perceptron (MLP) is capable of predicting the five day percentage change in the value of the FTSE 100 Index. While a strict interpretation of the weak and semi-strong forms of market efficiency would suggest the futility of attempting to generate abnormal risk-adjusted returns from publicly available information, a number of studies have cast doubt on the premise of monolithic market efficiency. Claimed anomalies in the EMH include the suggestion that markets do not always impound new information instantaneously (Hong, Lim and Stein, 2000), that stock markets can overreact as a result of excessive investor optimism or pessimism (Dissanaike, 1997) and that returns on the market are related to the day of the week or the month of the year (DeBondt and Thaler, 1987). Interpretation of the results of these studies has been controversial (Fama, 1998) but they are consistent with a hypothesis that market efficiency is a relative term. Under this premise, as market participants uncover new information processing mechanisms, market efficiency is enhanced. This paper implicitly takes this perspective.

Although there is a growing empirical literature describing the application of neural network methodologies to the modelling of stock market indices (Fernandez-Rodriguez, Gonzalez-Martel and Sosvilla-Rivero, 2000), differences in both the microstructures of equity markets and the way individual market indices are calculated, imply that research findings from one market may not transfer to another. Much work to date modelling market indices, has concentrated on US equity markets. The majority of studies develop a single neural network model

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 137–143, 2002.

and make no attempt to assess whether the quality of the model's predictions degrades over the out-of-sample testing period.

The remainder of this paper is organised as follows. Section 2 provides a brief overview of the MLP. Section 3 discusses the methodology and the data used in this study. The results are presented in Sect. 4 and conclusions are discussed in section 5.

2 Overview of the MLP

A brief description of the neural network utilised in this paper, a fully-connected, feedforward MLP, follows. An MLP generally consists of a collection of simple non-linear processing elements which are linked in a node-arc structure. This linkage of simple elements gives rise to emergent capabilities for the network, permitting complex input-output mappings. Nodes may serve as holding points for model inputs (input nodes), as holding points for the model's output (output node) or act as a processing unit, receiving signals from nodes, and in turn producing an output which is transmitted to other nodes. This signal can be modified (strengthened or weakened) when in transit along an interconnection (arc). In constructing an MLP the objective is to determine the appropriate node-arc structure and the appropriate arc weights which act to modify signals in transit between nodes. In mapping a set of input data to corresponding outputs, an MLP can be described as a generalised, semi-parametric, non-linear, regression model.

3 Methodology

The predictive horizon plays a significant role in determining the utility of potential input variables in index prediction. For example, many macroeconomic variables which may play a role in determining stock prices are reported infrequently and therefore are predominantly invariant in the context of a short-term prediction model[1]. The inputs used in the model were selected from a range of technical, fundamental and intermarket data suggested in prior literature (Chan, Jegadeesh and Lakonishok, 1996; Dissanaike, 1997; Brock, Lakonishok and LeBaron, 1992; Murphy, 1999). Initially, a series of technical indicators, drawn from price, volume and options data were calculated. These included a variety of moving averages, relative strength indicators, oscillators and lagged measures. An infinite number of technical indicators could be calculated based on historic index data. This study restricted attention to indicators calculated, within a twenty day period prior to the forecast date. In order to select the final subset of technical indicators included in the model, several data selection tools were employed, including correlation analysis, regression models and the construction of preliminary neural network models. A similar selection process

[1] Future expectations of these variables are not invariant but are difficult to observe.

was applied to select the intermarket and fundamental indicators. Ten inputs were included in the final model:

i. 5 day lagged percentage change in the value of the FTSE 100 index
ii. 20 day lagged percentage change in the value of the FTSE 100 index
iii. Ratio of the 10 vs 5 day moving average of the value of the FTSE 100 index
iv. Ratio of the 20 vs 10 day moving average of the value of the FTSE 100 index
v. Bank of England Sterling index
vi. S & P 500 composite index$_{t-1}$
vii. LIBOR 1 month deposit rate
viii. LIBOR 1 year deposit rate
ix. Aluminium ($ per tonne)
x. Oil ($ per barrel)

The data used in model development and testing was drawn from the period 1/1/92 to 31/12/96. Each model was developed using 918 days of trading data (793 training and 125 validation cases) and was tested out of sample on three subsequent six month periods (10/7/95–31/12/95, 1/1/96–28/6/96 and 1/7/96–31/12/96). Testing each six month period separately, facilitated the examination of the evolution of the predictive performance over time.

3.1 Model Selection

The final MLP model (11:6:1) utilised was as follows:

$$y_t = L\left(a_0 + \sum_{j=1}^{5} w_j L\left(\sum_{i=0}^{10} b_i w_{ij}\right)\right) \tag{1}$$

where b_i represents $input_i$ (b_0 is a bias node), w_{ij} represents the weight between input node$_i$ and hidden node$_j$, a_0 is a bias node attached to the output layer, w_j represents the weight between hidden node$_j$ and the output node and L represents the hyperbolic tangent transformation function. The size of the hidden layer was determined by trial and error, being selected on the basis of producing minimum prediction error, defined as root mean squared error (RMSE), on the validation data set.

3.2 Model Stacking

The majority of previous applications of neural network modelling methodology to index prediction consist of the construction of a single model. This approach implicitly assumes that there is a dominant global error minimum and implicitly hopes that the constructed model approaches this minimum. Given the limitations of a problem domain in which input / output relationships are dynamic and where input data is incomplete and noisy, the error surface may not have this property. In such a topology, no single model may be dominant and there may be potential to improve predictive quality by building multiple models.

This study constructs 25 separate models using both different starting points on the error surface and different randomisations of data between training and test datasets. In a similar concept to the combination of the output of simple non-linear processing elements in an individual distributed neural network, the predictions of individual neural network models are combined (*stacked*) to form a committee decision. The overall output of a stacked neural network is a weighted combination of the individual network outputs:

$$F(in) = \sum_{i=1}^{n} w_i f_i(in) \tag{2}$$

where F*(in)* is the output from the stacked network for input vector *(in)*, $f_i(in)$ represents the output of $network_i$ and w_i represents the stacking weight. In this study, the predictions are combined on an equally weighted linear basis. Thus, in calculating the predicted 5 day percentage change in the market index, the average prediction of all 25 models is used.

The predictive ability of an individual neural network is critically impacted by the choice of network weights. Since the back-propagation algorithm is a local search technique, the initial weights can have a significant impact on the solution chosen by the neural net. The stacking process is introduced both to reduce this problem and to combine the predictive abilities of individual models which may possess differing pattern recognition capabilities. The decision to select 25 models for stacking purposes is guided by the findings of Zhang, Martin, Morris and Kiparissides (1997) which suggest that stacking 20 to 30 networks is usually sufficient to stabilise model errors. This approach reduces the dependence of the final prediction on initial conditions stemming from the order of input/output data vectors in the dataset used for training and validation purposes and bears similarities to the bootstrap technique, in that repeated, random samples are drawn from the initial training / validation dataset. A number of studies have previously applied variants on this methodology (Zhang, Martin, Morris and Kiparissides, 1997; Franke and Neumann, 2000; Zhang, 1999) but as yet, few studies, other than Yoda (1994) have applied the methodology in the finance domain. Yoda (1994), in a study of the Tokyo stock market, found that the averaging of predictions resulted in enhanced predictive accuracy.

4 Results and Discussion

This section outlines the results obtained in each of the out-of-sample test periods and discusses the stability of the quality of the models' predictions over time.

4.1 RMSE and Correlation

Tables 1 and **2** summarise the results for each out-of-sample test period. **Table 1** shows the average RMSE, calculated as the average of the RMSEs for each of the 25 model's individual daily predictions. This is contrasted with the

RMSE of the average prediction of all 25 models taken together as presented in **Table 2**.

Table 1. Average RMSE of each model's predictions

	10/7/95-31/12/95	**1/1/96-28/6/96**	**1/7/96-31/12/96**
RMSE	0.2335	0.2754	0.3598

Table 2. RMSE of combined prediction

	10/7/95-31/12/95	**1/1/96-28/6/96**	**1/7/96-31/12/96**
RMSE	0.1833	0.1736	0.2143

The averaging of predictions across 25 networks has resulted in a noticeably lower error. It is also noteworthy that the RMSE has generally increased over the three time periods, indicating that the predictive quality of the models is degrading over time. To investigate this point further, the correlation co-efficients between the model's (models') predicted output and the actual percentage five day index change were calculated. These are summarised in **Tables 3** and **4**:

Table 3. RMSE of combined prediction

	10/7/95-31/12/95		**1/1/96-28/6/96**		**1/7/96-31/12/96**	
	Pearson	*Spearman*	*Pearson*	*Spearman*	*Pearson*	*Spearman*
r	0.2830	0.2833	0.2009	0.2217	0.1669	0.1689

Table 4. Correlation actual vs predicted 5 day change: combined prediction

	10/7/95-31/12/95		**1/1/96-28/6/96**		**1/7/96-31/12/96**	
	Pearson	*Spearman*	*Pearson*	*Spearman*	*Pearson*	*Spearman*
r	0.3973	0.4049	0.2902	0.3028	0.2231	0.2293

The predictions made as a result of averaging the predictions over 25 models show a higher correlation to the actual five day change than do the average correlations of the individual models[2]. In both cases, the correlation coefficients show degradation over time.

[2] For comparison purposes, a linear regression model was fitted to the training dataset. When applied to the first out-of-sample dataset, it resulted in a RMSE of 0.7417 and a Pearson correlation of 0.0023.

5 Conclusions

The aim of this study was to determine whether a neural network model developed using data drawn from the period 1/1/92 to 9/7/95 could predict the five day percentage change in the value of the FTSE 100 Index during the period 10/7/95 to 31/12/96, and to examine the utility of employing a stacking methodology. The results of the study suggest that neural network models can be constructed which have predictive ability, that the structure detected by the models is persistent, and that in the absence of a major market shock, predictive quality degrades gracefully. The results also suggest that a stacking methodology can improve predictive quality. The apparent ability of the non-linear neural networks to detect structure in past index, intermarket and fundamental data could arise for a number of reasons. The structure could be apparent, merely resulting from a statistical anomaly or it could be real, suggesting that market movements are non-random. Given the dynamic nature of the stock market, it is not possible to conclusively disprove the first claim based on the results of a single empirical study.

Acknowledgment. The author wishes to thank the anonymous reviewers for their valuable comments.

References

1. Brock, W., Lakonishok, J. and LeBaron B. (1992). 'Simple Technical Trading Rules and the Stochastic Properties of Stock Returns', *Journal of Finance*, 47(5):1731–1764.
2. Chan, L. K. C., Jegadeesh, N. and Lakonishok, J. (1996). 'Momentum strategies', *Journal of Finance*, 51(5):1681–1714.
3. DeBondt, W. and Thaler, R. (1987). 'Further Evidence on Investor Overreaction and Stock Market Seasonality', *Journal of Finance*, 42(3):557–581.
4. Dissanaike, G. (1997). 'Do stock market investors overreact?', *Journal of Business Finance & Accounting (UK)*, 24(1):27–50.
5. Fama, E. (1998). 'Market efficiency, long-term returns, and behavioral finance', *Journal of Financial Economics*, 49(3):283–306.
6. Fernandez-Rodriguez, F., Gonzalez-Martel, C. and Sosvilla-Rivero, S. (2000). 'On the profitability of technical trading rules based on artificial neural networks: Evidence from the Madrid stock market', *Economics letters*, 69:89–94.
7. Franke, J. and Neumann, M. (2000). 'Bootstrapping Neural Networks', *Neural Computation*,12:1929–1949.
8. Hong, H., Lim, T. and Stein J. (2000). 'Bad News Travels Slowly: Size, Analyst Coverage, and the Profitability of Momentum Strategies', *Journal of Finance*, 55(1):265–295.
9. Murphy, John J. (1999). *Technical Analysis of the Financial Markets*, New York: New York Institute of Finance.
10. Yoda, M. (1994). 'Predicting the Tokyo Stock Market'; in Deboeck, Guido J. (Editor) *Trading on the Edge: Neural, Genetic and Fuzzy Systems for Chaotic Financial Markets*, New York: John Wiley and Sons Inc.

11. Zhang, J., Martin, E.B., Morris, A.J. and Kiparissides C. (1997). 'Inferential Estimation of Polymer Quality Using Stacked Neural Networks', *Computers and Chemical Engineering*, 21(Supplement):1025–1030.
12. Zhang, J. (1999). 'Developing robust non-linear models through bootstrap aggregated neural networks', *Neurocomputing*, 25:93–113.

Experiments in Sparsity Reduction: Using Clustering in Collaborative Recommenders

Derek Bridge and Jerome Kelleher

University College, Cork

Abstract. The high cardinality and sparsity of a collaborative recommender's dataset is a challenge to its efficiency. We generalise an existing clustering technique and apply it to a collaborative recommender's dataset to reduce cardinality and sparsity. We systematically test several variations, exploring the value of *partitioning* and *grouping* the data.

1 Collaborative Recommenders

Given a set of items $i = 1 \ldots m$ and a set of users $u = 1 \ldots n$, a collaborative recommender will predict the rating $p_{au,ai}$ that active user au will assign to an item ai using the ratings that the other users have assigned to the items $r_{u,i}$.

Our experiments use the MovieLens dataset (`www.grouplens.org`). In this dataset, there are 100000 ratings by 943 users on 1682 items (movies) and the data has been cleaned up to include only users who have rated at least 20 movies. This dataset has been the subject of an extensive empirical investigation [2]. We can adopt the design decisions that [2] shows to give the best results.

Specifically then, our collaborative recommenders will work as follows:

- The similarity $w_{au,u}$ between the active user au and each other user u is computed using Pearson Correlation [2]. Following [2], the Pearson Correlation is weighted by a factor of $\frac{s}{50}$ where s is the number of co-rated items, to decrease similarity between users who have fewer than 50 co-rated items.
- After computing the similarity between au and each other user u, the 20 nearest neighbours are selected, i.e. the 20 for whom $w_{au,u}$ is highest.
- The predicted rating $p_{au,ai}$ is computed from the neighbours' ratings of ai as follows:

$$p_{au,ai} \hat{=} \bar{r}_{au} + \frac{\sum_{u=1}^{k}(r_{u,ai} - \bar{r}_u)w_{au,u}}{\sum_{u=1}^{k} w_{au,u}} \quad (1)$$

  where k is the number of neighbours (in our case, 20). This is essentially a weighted average of the neighbours' ratings for ai (weighted by their similarities). But it normalises the ratings (by using deviations from mean ratings) to counter the effects of 'niggardly' or 'immoderate' users.

In collaborative systems, many possible items will be available to many users, most of which the users will not have rated. The high cardinalities give efficiency problems but also, if ratings are too sparse, the prediction accuracy can be lower than that of a non-personalised recommender [2][3]. We investigate one possible solution to these problems: clustering the dataset.

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 144–149, 2002.

2 The Clustering Algorithm

The clustering algorithm which we have chosen is used in [1] to cluster users. The algorithm recursively invokes the k-means clustering algorithm.

In the k-means algorithm, elements are repeatedly assigned to clusters on the basis of their similarity to the cluster centre. We use Pearson Correlation again to compute similarity. The centres are initially seeds, provided as parameters. As the algorithm iterates, the centres become the centroids of the existing clusters, where a centroid is a new 'dummy' element formed by averaging ratings within the cluster. We consider the algorithm to have converged when the size of all clusters is the same on two successive iterations. We found empirically that convergence usually takes 15 iterations. To ensure that the algorithm always terminates, even on pathological data, we imposed a complementary stopping criterion in the form of an iteration limit of 20.

The RecTree clustering algorithm [1] calls k-means to split successive datasets into child clusters. It returns a binary tree of clusters, where the root represents the whole dataset. The algorithm splits the dataset if its size is greater than the maximum cluster size. It selects seeds using the 'two mutually well separated centres policy' [1]. The first seed is the element within the dataset whose distance is greatest from the dataset centroid. The second seed is the element which is most distant from the first seed.

The algorithm terminates when the sizes of all leaf nodes are less than or equal to the maximum cluster size and so no more subdivision is required. In some cases however, when data is not suitably distributed, growth in the tree can be 'lopsided'. This is where a very small and a very large cluster are created at each invocation. In this case we prevent the construction algorithm from over-partitioning the data by limiting the number of internal nodes (in our case to 2000). Because of this the algorithm makes no guarantee about the size of any leaf cluster. If the algorithm terminates normally, all leaf clusters will have at most the maximum cluster size number of elements. Of course, some might have very few members; if they have fewer than a certain threshold (in our case 10), we call the node an *outlier node*.

3 The Role of Clustering

We have developed and experimented with four collaborative recommender systems. In this paper, only three of the systems will be described, although the graphs show the results for all four systems. Where clustering is used in different systems, it is used for different purposes: *partitioning* and *grouping*.

Naive. Our Naive system acts as a baseline comparison system. It operates as described in Sect. 1: it exhaustively computes all similarities, picks the 20 nearest neighbours, and computes a prediction from these.

Table 1. Clustering items into superitems

	110	111	112	113	114	115	116	117
Jim	3		3				3	
Kelly		2	5		3		2	
Ray					4	4		3
Kim	4		3				2	

(a) Raw data

	S1={110,111,112}	S2={113,114,115}	S3={116,117}
Jim	3		3
Kelly	3.5	3	2
Ray		4	3
Kim	3.5		2

(b) Schematic illustration of the effect of item clustering

Clustered Users. We use clustering to form *partitions* of users. This reduces the *effective* cardinality of the dataset as only users from within a particular partition are considered as potential predictors. To produce a prediction for au, we apply the Naive algorithm only to the partition containing au. However, if a user is a member of an outlier node (Section 2), then we ascend up the Rec-Tree and use the value from the centroid of the parent. In effect what we are doing here is using a non-personalised recommendation in cases where we do not have enough cluster members to provide an accurate collaborative recommendation.

Clustered Items. We use clustering to form *groups* of items ('superitems'). A superitem's rating will be the average of its member ratings. By grouping items into superitems, we reduce the cardinality, and hence the sparsity, of the dataset; see Tables 2(a) and 2(b). To cluster items we use Pearson Correlation but we compare items over co-rating users (instead of comparing users over co-rated items). Once we have clustered items in the dataset into superitems, we can produce predictions. To predict Jim's rating for item 113, for example, we find the superitem to which 113 belongs, S2, and then we make a prediction for S2 by applying the Naive algorithm to the superitems data.

4 Experimental Methodology

In each experiment, the MovieLens dataset is split into two disjoint sets, the training set (80%) and the test set (20%). All results are subject to five-fold cross validation, each time using a different 80/20 split.

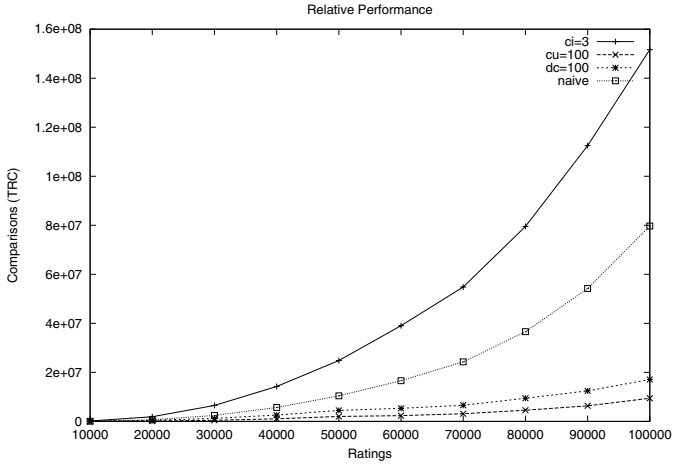


Fig. 1. Efficiency results

To give an implementation-independent measure of *efficiency*, we count Total Rating Comparisons (TRC), i.e. the number of rating comparisons carried out by the system in making a prediction. To measure prediction *accuracy*, we use Mean Absolute Error (MAE) (the average of the absolute difference between the system's predicted rating and the user's actual rating). *Coverage* is measured by counting the number of times the system fulfils a prediction request and reporting this as a percentage of the overall number of prediction requests made.

One parameter of the experiments that we had to set was the maximum cluster size. To estimate the best values to use for this parameter, we ran some experiments in which we varied the maximum cluster sizes as we increased the number of ratings from 10000 to 100000. There is no room to plot or discuss the results from these experiments here, but we summarise our findings below.

For *Clustered Users*, we found that a maximum cluster size of 100 users gave a good trade-off between accuracy and running time; in our graphs this system is denoted by the line cu = 100. For *Clustered Items*, we found that, of the sizes tried, a maximum cluster size of 3 was consistently best for efficiency, accuracy and coverage; in our graphs this system is denoted by the line ci = 3.

5 Results

5.1 Efficiency (Fig. 1)

Clustered Users carries out the lowest number of comparisons. The reason for this is the reduced number of potential advisors for each user. This gives a significant improvement over Naive in which all users in the system are potential advisors.

Clustered Items is by far the least efficient. This may seem counter-intuitive, but it is explained by realising that grouping items decreases sparsity, which

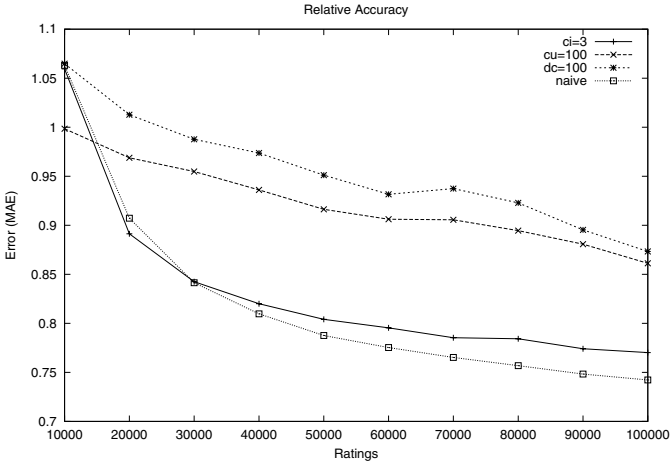


Fig. 2. Accuracy results

increases the number of co-rated (super)items to be compared. This is shown in Tables 2(a) and 2(b) where we reduce the cardinality from eight to three by clustering on contiguous item identifiers. Suppose we want to predict Jim's rating for item 117. In the raw data, Ray is the only person in a position to provide a prediction. But, when we produce a prediction for Jim using the clustered items, we find that item 117 is a member of cluster (superitem) S3. All of the other users have rated item S3 (because they all rated one or more of its constituent items), so we must consider each of these other users as a predictor, so we must compare Jim for similarity with *three times* more users.

5.2 Accuracy (Fig. 2)

Naive is generally the most accurate of the systems as it always finds the optimal advisors for each request. But, when the number of ratings in the system is small, Clustered Users has the lowest error. This reflects a well known property of collaborative recommender systems: when the number of ratings is low, a non-personalised recommender is more accurate than a collaborative one. Recall that Clustered Users resorts to non-personalised recommendations for outlier nodes. However, the accuracy of Clustered Users is soon surpassed by Naive and Clustered Items, because the optimal advisors for a prediction on an item are not generally members of the same partition as the active user. Clustered Items is almost as accurate as Naive, but some additional error is introduced due to inappropriately clustered items.

5.3 Coverage (Fig. 3)

Clustered Users has by far the worst coverage. By partitioning users, we are making useful ratings inaccessible to users in other partitions. Clustered Items

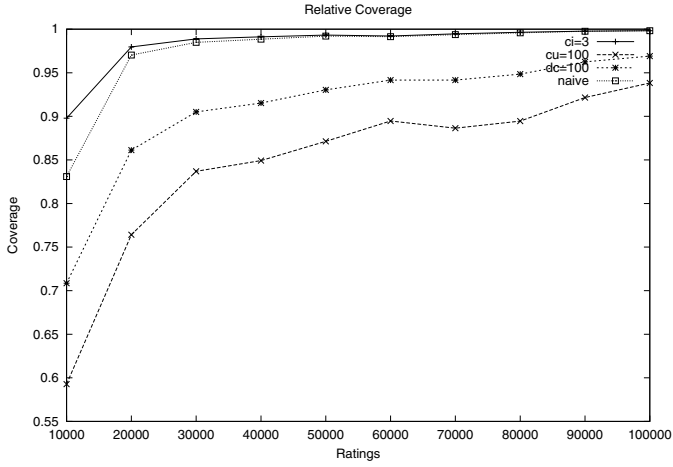


Fig. 3. Coverage results

has very good coverage, sometimes higher than Naive. This is easily explained by an example. In Table 2(a), we cannot obtain a prediction for Kim on item 113: no-one has rated item 113. However, in Table 2(b), item 113 is a member of cluster S2. Both Kelly and Ray have ratings for superitem S2, so we can provide a prediction for Kim on superitem S2, and hence for item 113.

6 Conclusions

We presented results from some of our experiments on different systems. The *partitioning* given by Clustered Users results in by far the best efficiency. Accuracy is highest in the Naive system as it always finds optimal advisors for every prediction request, but the *grouping* given by Clustered Items yields surprisingly low error. Clustered Users suffers a severe accuracy penalty as the partitioning distributes item ratings unevenly among the partitions. Clustered Items has the further advantage of increasing coverage in cases where data is very sparse but the efficiency implications of this approach are very severe.

References

1. Chee, S.H.S.: *RecTree: A Linear Collaborative Filtering Algorithm*, M.Sc. Thesis, Simon Fraser University, 2000.
2. Herlocker, J.L.: *Understanding and Improving Automated Collaborative Filtering Systems*, Ph.D. Thesis, University of Minnesota, 2000.
3. Resnick, P., N. Iacovou, M. Suchak, P. Bergstrom & J. Riedl: GroupLens: An Open Architecture for Collaborative Filtering of Netnews, in *Procs. of ACM CSCW'94 Conference on Computer-Supported Cooperative Work*, pp. 175–186, 1994.

Identification of Visual Features Using a Neural Version of Exploratory Projection Pursuit

Emilio Corchado[1] and Colin Fyfe[2]

[1] Area de Lenguajes y Sist. Informáticos .
Departamento de Ingenieria Civil
Universidad de Burgos. Spain
Tel. +34 947 25 8989 Fax +34 947 25 8910
escorchado@ubu.es

[2] Applied Computational Intelligence Research Unit
The University of Paisley. Scotland
Tel: +44 141 848 3305 Fax: +44 141 848 3542
colin.fyfe@paisley.ac.uk

Abstract. We develop artificial neural networks which extract structure from visual data. We explore an extension of Hebbian Learning which has been called $\mathcal{E}$ - Insensitive Hebbian Learning and show that it may be thought of as a special case of Maximum Likelihood Hebbian learning and investigate the resulting network with both real and artificial data. We show that the resulting network is able to identify a single orientation of bars from a mixture of horizontal and vertical bars and also it is able to identify local filters from video images.

1 Introduction

This paper investigates the problem of how brains can find structure in their visual environment: there are a great number of different visual stimuli which an organism requires to identify and classify and this seems to begin to happen at a relatively early stage in visual processing. This problem has proved to be rather difficult for traditional computer programs yet it is a problem which every living sighted organism learns to master relatively early in life. This suggests that there is something in the hardware on which they run – the neurons and synapses – which make this difficult problem tractable. Thus we develop an artificial neural network which working on raw visual information alone extracts interesting structure from the data in a fast and efficient manner. The network is linked to the statistical technique of Exploratory Projection Pursuit.

2 The Negative Feedback Network

First we introduce the Negative Feedback Network [9]. Feedback is said to exist in a system whenever the output of an element in the system influences in part the input

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 150-157, 2002.

applied to that particular element. It is used in this case to maintain the equilibrium on the weight vectors.

Consider an N-dimensional input vector, $\mathbf{x}$, and a M-dimensional output vector, $\mathbf{y}$, with W_{ij} being the weight linking input j to output i and let η be the learning rate. The initial situation is that there is no activation at all in the network. The input data is fed forward via weights from the input neurons (the $\mathbf{x}$ -values) to the output neurons (the $\mathbf{y}$ -values) where a linear summation is performed to give the activation of the output neuron. We can express this as:

$$y_i = \sum_{j=1}^{\mathbf{N}} W_{ij} x_j, \forall i \tag{1}$$

The activation is fed back through the same weights and subtracted from the inputs (where the inhibition takes place):

$$e_j = x_j - \sum_{i=1}^{M} W_{ij} y_i, \forall j \tag{2}$$

After that simple Hebbian learning is performed between input and outputs:

$$\Delta W_{ij} = \eta e_j y_i \tag{3}$$

The effect of the negative feedback is to stabilise the learning in the network. Because of that it is not necessary to normalise or clip the weights to get convergence to a stable solution.

Note that this algorithm is clearly equivalent to Oja's Subspace Algorithm [20] since if we substitute equation (2) in equation (3) we get:

$$\Delta W_{ij} = \eta e_j y_i = \eta \left(x_j - \sum_k W_{kj} y_k \right) y_i \tag{4}$$

This network is capable of finding the principal components of the input data [10] in a manner that is equivalent to Oja's Subspace algorithm [20], and so the weights will not find the actual Principal Components but a basis of the Subspace spanned by these components.

Since the model is equivalent to Oja's Subspace algorithm, we might legitimately ask what we gain by using the negative feedback in this way. Writing the algorithm in this way gives us a model of the process which allows us to envisage different models which would otherwise be impossible ([15,12]).

Factor Analysis[4] is a technique similar to PCA in that it attempts to explain the data set in terms of a smaller number of underlying factors. However Factor Analysis begins with a specific model and then attempts to explain the data by finding parameters which best fit this model to the data. [3] have linked a constrained version of the Negative Feedback network to Factor Analysis.

3 An Extension of Hebbian Learning

The nonlinear PCA rule [23]:

$$\Delta W_{ij} = \eta\left(x_j f(y_i) - f(y_i)\sum_k W_{kj} f(y_k) \right) \tag{5}$$

can be derived as an approximation to the best non-linear compression of the data.

Thus we may start with a cost function

$$J(W) = 1^T E\left\{ \left(\mathbf{x} - Wf\left(W^T\mathbf{x}\right)\right)^2 \right\} \tag{6}$$

which we minimise to get the rule(5).[16]used the residual in the linear version of (6) to define a cost function of the residual

$$J = f_1(\mathbf{e}) = f_1(\mathbf{x} - W\mathbf{y}) \tag{7}$$

where $f_1 = \|.\|^2$ is the (squared) Euclidean norm in the standard linear or nonlinear PCA rule. With this choice of $f_1(\ )$, the cost function is minimized with respect to any set of samples from the data set on the assumption that the residuals are chosen independently and identically distributed from a standard Gaussian distribution [2]. We may show that the minimization of J is equivalent to minimizing the negative log probability of the residual, $\mathbf{e}$, if e is Gaussian. Let:

$$p(\mathbf{e}) = \frac{1}{Z}\exp(-\mathbf{e}^2) \tag{8}$$

The factor Z normalizes the integral of $p(\mathbf{y})$ to unity.

Then we can denote a general cost function associated with this network as

$$J = -\log p(\mathbf{e}) = (\mathbf{e})^2 + K \tag{9}$$

where K is a constant. Therefore performing gradient descent on J we have

$$\Delta W \propto -\frac{\partial J}{\partial W} = -\frac{\partial J}{\partial \mathbf{e}}\frac{\partial \mathbf{e}}{\partial W} \approx \mathbf{y}(2\mathbf{e})^T \tag{10}$$

where we have discarded a less important term (see [19] for details).

In general [21] , the minimisation of such a cost function may be thought to make the probability of the residuals greater dependent on the pdf of the residuals. Thus if the probability density function of the residuals is known, this knowledge could be used to determine the optimal cost function.

[16] investigated this with the (one dimensional) function:

$$p(\mathbf{e}) = \frac{1}{2+\varepsilon}\exp\left(-|\mathbf{e}|_\varepsilon\right) \tag{11}$$

where:

$$|e|_\varepsilon = \begin{cases} 0 & \forall |\mathbf{e}| < \varepsilon \\ |\mathbf{e}| - \varepsilon & \text{otherwise} \end{cases} \tag{12}$$

with ε being a small scalar ≥ 0.

[16] described this in terms of noise in the data set. However we feel that it is more appropriate to state that, with this model of the pdf of the residual, the optimal $f_1(\ )$ function is the ε-insensitive cost function:

$$f_1(\mathbf{e}) = |\mathbf{e}|_\varepsilon \tag{13}$$

In the case of the Negative Feedback Network, the learning rule is

$$\Delta W \propto -\frac{\partial J}{\partial W} = -\frac{\partial f_1(\mathbf{e})}{\partial \mathbf{e}} \frac{\partial \mathbf{e}}{\partial W} \tag{14}$$

which gives:

$$\Delta W_{ij} = \begin{cases} 0 & \text{if } |e_j| < \varepsilon \\ \eta y_i (sign(e_j)) & \text{otherwise} \end{cases} \tag{15}$$

This change from viewing the difference after feedback as simply a residual rather than an error will permits us later to consider a family of cost functions each member of which is optimal for a particular probability density function associated with the residual.

4 A Neural Implementation of Exploratory Projection Pursuit

The ε-insensitive learning rule is clearly a member of the family of learning rules which are suggested by the family of exponential distributions. Let the residual after feedback have probability density function

$$p(\mathbf{e}) = \frac{1}{Z} \exp(-|\mathbf{e}|^p) \tag{16}$$

Then we can denote a general cost function associated with this network as

$$J = -\log p(\mathbf{e}) = |\mathbf{e}|^p + K \tag{17}$$

where K is a constant. Therefore performing gradient descent on *J* we have

$$\Delta W \propto -\frac{\partial J}{\partial W} = -\frac{\partial J}{\partial \mathbf{e}} \frac{\partial \mathbf{e}}{\partial W} \approx y(p\,|\mathbf{e}|^{p-1}\ sign(\mathbf{e}))^T \tag{18}$$

where T denotes the transpose of a vector. We would expect that for leptokurtotic residuals (more kurtotic than a Gaussian distribution), values of p<2 would be appropriate, while for platykurtotic residuals (less kurtotic than a Gaussian), values of p>2 would be appropriate. It is a common belief in the ICA community [18] that it is less important to get exactly the correct distribution when searching for a specific source than it is to use a model with an approximately correct distribution i.e. all supergaussian signals can be retrieved using a generic leptokurtotic distribution and all subgaus-

sian signals can be retrieved using a generic platykurtotic distribution. Our experiments will tend to support this belief to some extent but we often find accuracy and speed of convergence are improved when we are accurate in our choice of p.

Therefore the network operation is: feedforward (1), feedback(2) and:

(19) Weight change: $$\Delta W_{ij} = \eta . y_i . sign\left(e_j\right) | e_j |^{p-1}$$

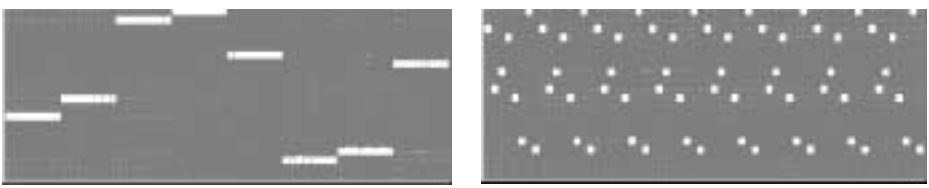
[16] described their rule as performing a type of PCA, but this is not strictly true since only the original (Oja) ordinary Hebbian rule actually performs PCA. We may link the method to the standard statistical method of Exploratory Projection Pursuit (EPP) [8,11,5]: EPP also gives a linear projection of a data set but choses to project the data onto a set of basis vectors which best reveal the interesting structure in the data; interestingness is usually defined in terms of how far the distribution is from the Gaussian distribution.

5 Experiments

5.1 Results Using an Artificial Data Set

The input data set is a 8x8 square grid which contains a mixture of horizontal and vertical bars [7]. There are (nearly) 2^{16} possible patterns in the data set. Each of the 16 possible bars is chosen with a fixed probability of 1/8 independent of each other. The bars are the underlying causes of the input data. The Factor Analysis network has been shown [13] to be capable of identifying just one of the horizontal or vertical bars at each of the network's output neurons.

In these experiments the number of outputs, which is 20, is greater than the number of bars, which is 16. The value of ε is 0.06, the learning rate is 0.01 and the number of iterations is 50000. Noise has not been added.



a) horizontal bars b) vertical bars

Fig. 1. The network is able to identify just one orientation of bars when p=1

A way to suppress one orientation of bars is just by changing the p parameter when the rest of the parameters keep the same value. The suppression of one bar orientation (Fig. 1) is achieved when p=1, which corresponds to ε -Insensitive learning rule. This is more appropriate for data sets which are kurtotic. For p=2, the network recognised

the 16 bars. For p greater than 2 some of the bars are recognised by more than one weight vector. p=2 is most appropriate for gaussian data. Now [14] showed that this data set is inherently 15 dimensional since each bar can be represented as a linear sum of the other 15 bars. Therefore representing this data set with 16 basis vectors means that we are using an overcomplete basis. Indeed [14] showed that the addition of noise on the outputs can be used to create a Minimum Overcomplete Basis (MOB) – the smallest number of basis vectors which can represent the underlying causes of the data set, the bars, without breaking any bar. In the case of p=1, which is appropriate for positively kurtotic data, the residual is either a whole bar of the opposite orientation to that in the data presented or is 0. In other words, we get a lot of residuals which are close to zero and a lot of residuals which are very large. This is a prescription for a kurtotic distribution and so only a single orientation is found by the network. For p greater than 2, the network is aiming to maximise the probability that the residuals are from a distribution with negative kurtosis which could be thought of as a distribution which is rather more uniform (or even bimodal) compared with a Gaussian. In this case sometimes two neurons respond to one bar. So the coding is more spread out and the residuals are liable to be more uniform than would be the case when each bar is identified by a single neuron.

5.2 Results Using Real Data

To test the network we use real data comprising different movies [17]of real life scenes. [14] used video clips from a great number of different sources. Thus movement in these images could be due to camera panning, movement of the targets, zooming of the camera etc. Our movies have been produced in such a way that single-cause movements have been captured. In Fig. 2, we show the type of filters found when we use p=3. They are local in time because for instance, in Fig. 2 we see zero values for the weights into the last 9 time slices and only non-zero values in the first three times; they are local in space because, in those three time slices, only the upper middle weights are non-zero.

Visual data [6] is characterised by elements of positive kurtosis. For example, [1] show that edges in images tend to have positively kurtotic distributions. Therefore our first interesting investigation is into what happens when our network is aiming to maximise the probability of residuals under a negatively kurtotic distribution. Now it is not possible, in general, to find negative kurtosis in the data set and so the network's best response is to remove any positive kurtosis in the data and have the residuals more close to Gaussian. Thus, we see (Fig. 2) that for values of p>2, we achieve a sparse response on the outputs.

We perform a rectification on the weights, $w = [w]^+$, and on the outputs, $y = [y]^+$. Thus if the weight update resulted in negative weights, those weights were set to zero; if the feedforward mechanism gives a negative output, this was set to zero. We will use the notation $[t]^+$ for this rectification: if t<0, t is set to 0; if t>0, t is unchanged.

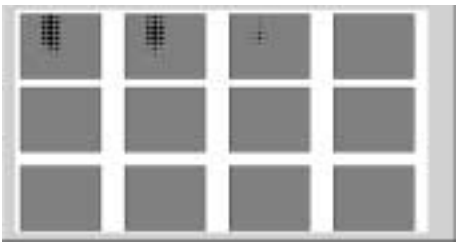


Fig. 2. The figure shows a weight vector. This rectangular box shows the weight vector connected to a different output neuron. The sizes of the weights are represented by the diameter of the small black circles (black represents a positive value). The square boxes within each rectangular window coincide with the weights connecting the output to an input from each of the 12 sequential images from the movie

Similar results were achieved with the other movies, with the shape of the individual filters determined by the data set.

6 Conclusions

We have shown that an unsupervised ANN using Maximum Likelihood Hebbian learning rules is capable of identifying the underlying causes in an artificial data and suppressing one orientation of bar.

An important finding achieved in this set of experiments is the suppression of one orientation by using the appropriated value for the parameter p in the learning rule. This network is able to identify local filters from video images

All networks used in this paper are biologically plausible in that they are based on the simple Hebbian rule.

References

1. H.B. Barlow,. and, D.J. Tolhurst. Why Do You Have Edge Detectors, Optical Society of America, Technical Digest, 23, 171. (1992).
2. C.M. Bishop. Neural Networks for Pattern Recognition, Oxford, 1995.
3. D. Charles, and C. Fyfe. Modelling Multiple Cause Structure Using Rectification Constraints. Network: Computation in Neural Systems, 9:167–182 , 1998.
4. D. Charles, and C. Fyfe, Rectified Gaussian Distribution and the Identification of Multiple Causes Structure in Data. ICANN 99, 1999.
5. E. Corchado, D. MacDonald and C. Fyfe, Maximum and Minimum Likelihood Hebbian Learning. Data Mining and Knowledge Discover. Submitted. 2002.
6. D.J. Field, What is the goal of sensory coding? Neural Computation 6, 559–60, 1994.
7. P. Földiák, Models of Sensory Coding PhD thesis, University of Cambridge, 1992.
8. J. Friedman, and J. Tukey, A Projection Pursuit Algorithm for Exploratory Data Analysis. IEEE Transaction on Computers, (23): 881–890, 1974.

9. C. Fyfe, “PCA Properties of Interneurons”, From Neurobiology to Real World Computing, Proceedings of International Conference on Artificial on Artificial Neural Networks, ICAAN 93, pages 183–188, 1993.
10. C. Fyfe, Negative Feedback as an Organising Principle for Artificial Neural Networks, PhD Thesis, Strathclyde University, 1995.
11. C. Fyfe, and R. Baddeley, Non-linear data structure extraction using simple Hebbian networks, Biological Cybernetics 72(6), p533–541, 1995.
12. C. Fyfe, A Scale Invariant Map. Network: Computing in Neural Systems, 7: pp 269–275, 1996.
13. C. Fyfe, A Neural Network for PCA and Beyond, Neural Processing Letters, 6:33–41, 1997.
14. D. Fyfe and C. Charles. Using Noise to Form a Minimal Overcomplete Basis, ICANN 99. 1999.
15. C. Fyfe and E. Corchado. Maximum Likelihood Hebbian Rules. European Symposium on Artificial Neural Networks. Esann 2002, 2002.
16. C. Fyfe, and D. MacDonald, ε-Insensitive Hebbian learning, Neuro Computing, 2001.
17. Y. Han, and C. Fyfe, A General Class of Neural Networks for P.C.A and F.A. Pages 158–163. Intelligent Data Engineering and Automated Learning. IDEAL 2000, 2000.
18. A. Hyvärinen, Complexity Pursuit: Separating interesting components from time series. Neural Computation, 13: 883–898, 2001.
19. J. Karhunen, and J. Joutsensalo, Representation and Separation of Signals Using Nonlinear PCA Type Learning, Neural Networks, 7:113–127, 1994.
20. E. Oja, Neural Networks, Principal Components and Subspaces, International Journal of Neural Systems, 1:61–68, 1989.
21. A.J. Smola, and. B. Scholkopf, A Tutorial on Support Vector Regression. Technical Report NC2-TR-1998-030, NeuroCOLT2 Technical Report Series. 1998.
22. J.H. Van Hateren, and A. Van der Schaaf, Independent Component Filters of Natural Images Compared with Simple Cells in Primary Visual Cortex. Proceedings of The Royal Society. London. B 265, 359–366, 1998.
23. L Xu. Least Mean Square Error Reconstruction for Self-Organizing Nets", Neural Networks, Vol. 6, pp. 627–648, 1993.

How People Compare an Item's Placement in Two Alternative Categories

Fintan Costello

School of Computer Applications, Dublin City University, Glasnevin, Dublin 9, Ireland
Fintan.Costello@compapp.dcu.ie

Abstract. This paper describes an experiment examining how people compare two alternative categories and decide whether a given item would be better placed in one category or the other. The experiment asks whether an item's comparative classification in one of two alternative categories is related to the difference between the item's typicality as a member of those two categories. The alternative categories compared in the experiment are single categories such as "bird" and combined categories such as "pet bird". The experiment found that an item's comparative classification in two such categories could not be predicted from the item's typicality in those categories: some items had similar typicality in both single and combined categories but were judged to be much better placed in the combined rather than the single category. This suggests that some other factors such as category specificity or informativeness may be involved in comparative classification.

1 Introduction

The ability to classify items and place them in categories is fundamental to human cognition and reasoning. An important part of classification is the ability to decide whether a given item would be better placed in one category or in an alternative, related category. The comparison of two alternative categories is seen in many situations; for example, when deciding whether a given crime is better placed in the category "robbery" or the related category "robbery with menaces", or whether a given patient is better classified under the diagnosis "pneumonia" or the related diagnosis "pneumonia and sinusitis". This paper describes an experiment on how people carry out comparative classification.

When asked to classify items in categories, people's responses are naturally graded: some items are rated as more typical and representative category members than others. Typicality is an important predictor of various aspects of classification: for example, the speed with which people classify an item in a category is proportional to the typicality of that item for that category, and the probability of someone making an error in classifying an item in a category decreases the more typical the item is for the category [1,2,3]. The current experiment asks whether an item's comparative classification in two alternative categories (the degree to which people judge the item would be better placed in one category rather than the other) is related to the difference between the item's typicality scores in the two categories being compared. If compara-

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 158-164, 2002.

tive classification is related to typicality, then a difference between an item's typicality scores in two categories should mean a comparative preference for one category over the other, and little difference in typicality scores should mean little comparative preference for one category over the other. If comparative classification is not related to typicality, no such relationship should be seen.

The pairs of categories used in the experiment consisted of a single familiar category (e.g. "bird") and a combined category containing that single category (e.g. "pet bird"). Pairs of this kind have been used in previous studies of classification and typicality [4,5,6,7]. Such pairs allow a wide range in typicality scores for items: some items are highly typical members of both categories in such pairs (e.g. parrots are rated as typical birds and equally typical "pet birds"), while other items are typical of one category but not of the other (robins are typical birds but not typical "pet birds"). This range of typicality scores allows the relationship between comparative classification and typicality to be examined for different typicality values.

2 Experimental Study

This study examines the relationship between typicality score and comparative category preference for items in single natural-language categories and in natural-language category combinations. In a typicality rating task, participants rated the typicality of items in single categories and combined categories independently, using a methodology from previous studies of typicality [4,5,6]. In this task, for example, participants would first be asked to rate the degree to which the term "bird" applies to the item "parrot"; later, they would be asked to rate the degree to which the combination "bird that is also a pet" applies to that item. In a comparative category rating task, a different group of participants rated their comparative preference for placing of the same items in the same single and combined categories. In this task the single and the combined category were presented together, and participants are asked to judge which applied better to a given item: so, for example, participants were asked to judge whether the term "bird" or the term "pet bird" applies better to the item "parrot", and to rate the degree to which one term applied better than the other. The same items, single categories and combinations were used in both tasks.

2.1 Method

Materials. The materials for this experiment were 6 single-category / relative-combination pairs, with 5 items for each pair (see Table 1). The category/combination pairs consisted of a single category such as "bird" and a relative-clause combination containing that category, such as "bird that is a pet". The category / combination pairs were chosen so that there would be some items which were good members of both the single category and the combination. The 5 items for each pair were in 5 different classes. Items were selected to vary in their typicality for the categories in question. Class (i) items were highly typical for both the single category and the combination;

class (ii) items were typical for the single category but less typical for the combination; class (iii) items were typical for the single category, but completely untypical for the combination; class (iv) items were less typical for the single category, and completely untypical the combination; and class (v) items were completely untypical for both the single category and the combination.

Table 1. Categories, category combinations, and items used in Experiment. Items grouped in 5 classes according to typicality for single categories and combinations (see text)

Category	Category combination	items				
		Class(i)	Class(ii)	Class(iii)	Class(iv)	Class(v)
Bird	Bird that is also a Pet	Parrot	Canary	Robin	Chicken	Pig
Utensil	Utensil that is also a Weapon	Knife	Fork	Ladle	Kettle	Radio
Utensil	Utensil that is also a Container	Cup	Dish	Spoon	Toaster	Window
Fish	Fish that is also a Pet	Goldfish	Guppy	Salmon	Crab	Snail
Building	Building that is also a Dwelling	House	Castle	Hotel	Bus-stop	Bicycle
Plant	Plant that is also a Food	Lettuce	Carrot	Tulip	Lichen	Granite

Participants. Participants were 24 undergraduate students at Dublin City University. Two blocks of 12 participants each were randomly assigned to the two different tasks.

Procedure. Participants in both conditions received questionnaire booklets which they were asked to complete. Each booklet began with an instruction sheet and continued with a series of rating questions for participants to answer. Participants were asked to answer every question. If they came across a word they did not understand, they were asked to circle it and skip the question.

Participants in the typicality rating task received an instruction sheet adapted from previous experiments [5,6]. Participants were asked to indicate how well a given term applied to a series of items by marking a 7-point scale going from -3 ("does not apply") to +3 ("applies very well"). The example term "a sport" was used to demonstrate the rating process, and illustrative ratings were given for the items "Soccer", "Skiing", and "Chess". Following the instruction sheet, participants in the typicality rating condition received 12 question sheets, each with a single category or a combination at the top and followed by the 5 items belong to that single category or combination. Next to each item was a scale on which participants rated how good the term at the top of the page applied to that item. Items on each sheet were in random order. The first 6 question sheets had single categories at the top; within those sheets categories were in random order. The following 6 sheets had combined categories at the top; within those, combinations were in random order.

Participants in the comparative category rating task received an instruction sheet explaining the comparative membership task in words similar to those used in the typicality rating instructions. The instruction sheet told participants that they would be given two different phrases (a single category and a combination) and that their

task was to indicate which phrase applied better to a series of items by marking a points on a graded 7-point scale going from the first phrase (the single category) on the left, to the second phrase (the combination) on the right. The rating process was demonstrated using the example phrases "a sport" and "a game that is also a sport", and illustrative ratings were given for the items "Soccer", "Skiing", and "Chess". Following the instruction sheet, participants in the comparative category rating task received 6 question sheets. Each sheet consisted of the 5 items associated with a category/combination pair. Beside each item was a scale for indicating whether the single category or the combination applied better to that item. Items on each sheet, and sheets in each booklet, were in random order.

Table 2. Percentage of single-category preferences in the comparative rating task and in the typicality rating task, for each class of item

Task	Single-Category Preferences by Item Class				
	Class(i)	Class(ii)	Class(iii)	Class(iv)	Class(v)
Comparative rating task	12.5%	27.8%	66.7%	58.3%	20.8%
Typicality rating task	4.2%	30.6%	87.5%	52.8%	5.6%

2.2 Results

Participants in both the typicality rating and the comparative category rating tasks had no difficulty with their tasks, completing their question booklets in approximately 10 minutes. Of a total of 720 rating questions in both tasks, 18 were unanswered. Some participants indicated that they did not understand the item "Guppy" (7 participants) or the item "Lichen" (4 participants). However, since eliminating the data for these two items had no effect on the results, these items were retained in analysis. The data obtained in the typicality rating task and the comparative rating task were analysed in two ways. The first analysis looked at people's preferences for single categories over combined categories (single-category preferences) in the comparative-rating task and the typicality-rating task. The second analysis looked at people's preferences for combined categories over single categories (combined-category preferences), and compared the occurrence of such preferences in the two tasks.

Results: Single-Category Preferences. For each item a count was obtained of the number of participants in the comparative task who rated that item as a better member of the single category than of the combination ("comparative single-category preferences"). A count was also obtained of the number of participants in the typicality task who gave that item a higher typicality score in the single category than in the combination ("typicality-based single-category preferences"). There were 134 comparative single-category preferences, and 130 typicality-based single-category preferences. A paired t-test across all items showed no statistical difference between the number of typicality-based single-category preferences (M=4.33, SD=4.29) and comparative single-category preferences (M=4.47, SD=3.32; $t(29)=0.34$, $P = .73$) that an item

received. There was a strong correlation between an item's typicality-based and its comparative single-category preferences ($r=0.87$, $p < .001$).

Table 2 shows how single-category preferences were distributed across the 5 different classes of items used in the experiment, in the two different rating tasks. The distribution was similar for both tasks, further supporting the observed correlation between single-category preferences in the comparative and typicality rating task. In both tasks, single-category preferences occurring most often for class (iii) and class (iv) items such as ""Robin" and "Chicken"; such items were rated as better placed in the single category "bird" in the comparative rating task, and were given higher typicality scores for the single category "bird" in the typicality rating task. The correlation between typicality-based and comparative single-category preferences suggests that differences in typicality scores reliably predict preferences for a single category over a combined category.

Table 3. Percentage of Combined-Category preferences in the comparative rating task and in the typicality rating task, for each class of item

Task	Combined-Category Preferences by Item Class				
	Class(i)	Class(ii)	Class(iii)	Class(iv)	Class(v)
Comparative rating task	84.7%	63.9%	30.6%	22.2%	18.1%
Typicality rating task	19.4%	26.4%	0.0%	9.7%	4.2%

Results: Combined-Category Preferences. For each item a count was obtained of the number of participants in the comparative rating task who rated that item as a better member of the combined category than the single category ("comparative combined-category preferences"). A count was also obtained of the number of participants in the typicality rating task who gave that item a higher typicality score in the combined category than the single category ("typicality-based combined-category preferences"). There were 158 comparative combined-category preferences; 158 cases in which a participant in the comparative task rated an item as a better member of the combined category than the single category. However, there were only 30 typicality-based combined-category preferences; only 30 cases in which a participant in the typicality task gave an item a higher typicality score in a combined category than in the related single category. A paired t-test across all items showed a reliable difference between typicality-based combined-category preferences ($M=1.43$, $SD=2.34$) and comparative combined-category preferences ($M=5.27$, $SD=3.69$; $t(29)=6.50$, $P < .0001$).

In the case of combined-category preferences, then, there was a significant difference between the two tasks. For example, 75% of participants in the comparative rating task judged that the class (i) item "parrot" would be much better placed in the category "pet bird" rather than in the category "bird" (mean rated preference $M=2.5$ on a scale with a maximum of 3). However, only 16% of participants in the typicality rating task gave the item "parrot" a higher typicality rating in the category "pet bird"

than in the category "bird". (Most participants judged the item to be equally highly typical of the categories "bird" and "pet bird")

This difference was particularly evident for class (i) items (items which, like "parrot", were highly typical for both the single category and the combination). Table 3 shows how combined-category preferences were distributed across the 5 different classes of items used in the experiment in the two rating tasks. In the comparative rating task each class (i) item was placed in the combined category rather than the single category by 84.7% of participants, on average. In the typicality rating task, however, those class (i) items were given a higher typicality rating in the combined category than in the single category by only 19.4% of participants, on average. When considering preferences for a combined category over a single category, it appears that differences in typicality scores are unrelated to comparative category preferences.

3 General Discussion

The above experiment asked whether an item's comparative rating in two categories (the degree to which people thought it was better to place the item in one category or the other) could be predicted from the difference between the item's typicality scores for the two categories. The results showed that, when an item was judged to be better placed in a single category rather than a combined category, that judgment could be predicted from the difference in the item's typicality scores for the two categories. However, when an item was judged to be better placed in the combined category rather than the single category, that judgment was unrelated to any difference in the item's typicality scores. These results suggest that, when deciding whether an item is better placed in one category or another, people do not simply compare the item's typicalities for the two categories. Some other factor also comes into play. This result is surprising, given typicality's importance in many different aspects of classification.

What might the additional factor in comparative classification be? In the experiment, this additional factor was particularly significant with items that were equally highly typical for both categories being compared (as with the item "parrot" and the categories "bird" and "pet bird"). It may be that, when typicalities are equally high, people prefer to combined categories over single categories because the combined category classification response contains the single category response. According to this proposal, both "bird" and "pet bird" are correct classifications for the item "parrot": people choose "pet bird" as their preferential classification because that choice also includes the classification "bird". Alternatively, it may be that when typicalities are equal, people prefer combined categories over single categories because combined categories are narrower and more informative. According to this proposal, people choose "pet bird" for classifying the item "parrot" because that choice conveys more about the item in question than the single category "bird" would convey. Finally, a third proposal might be that, when comparing an item's classification in two categories, people look for features that distinguish between the two categories and classify solely on the basis of those features. For example, a feature that distinguishes between "birds" and "pet birds" is that "pet birds" are usually kept in cages. On the ba-

sis of this feature, an item such as "parrot" would be preferentially placed in the category "pet bird" rather than the category “bird”. Further experimental work is needed to distinguish between these three possibilities.

References

1. Rosch, E. & Mervis, C.D.: Family resemblance studies in the internal structure of categories. Cognitive Psychology **7** (1975) 573–605
2. Rosch, E.: Principles of categorization. In: Rosch, E., Lloyd, B.B. (Eds.): Cognition and Categorization. Hillsdale, NJ: Erlbaum (1978) 27–48
3. Komatsu, L.K.: Recent views of conceptual structure. Psychological Bulletin **112(3)** (1992) 500–526
4. Chater, N., Lyon, K., & Myers, T.: Why are conjunctive categories overextended? Journal of Experimental Psychology: Learning, Memory & Cognition **16(3)** (1990) 497–508
5. Hampton, J.A.: Overextension of conjunctive concepts: Evidence of a unitary model of concept typicality and class inclusion. Journal of Experimental Psychology: Learning, Memory & Cognition **15** (1988) 55–71
6. Hampton, J.A.: Conceptual combination: Conjunction and negation of natural concepts. Memory & Cognition **25(6)** (1997) 888–909
7. Huttenlocher, J., Hedges, L.V.: Combining graded categories: Membership and typicality. Psychological Review **101** (1994) 157–165

Investigations into Market Index Trading Models Using Evolutionary Automatic Programming

Ian Dempsey[1], Michael O'Neill[1], and Anthony Brabazon[2]

[1] Dept. Of Computer Science And Information Systems
University of Limerick, Ireland
Michael.ONeill@ul.ie
[2] Dept. Of Accountancy, University College Dublin, Ireland
Anthony.Brabazon@ucd.ie

Abstract. This study examines the potential of an evolutionary automatic programming methodology to uncover a series of useful technical trading rules for the US S&P stock index. Index values for the period 01/01/1991 to 01/10/1997 are used to train and test the evolved rules. A number of replacement strategies, and a novel approach to constant evolution are investigated. The findings indicate that the automatic programming methodology has much potential with the evolved rules making gains of approximately 13% over a 6 year test period.

1 Introduction

The objective of this study is to determine whether an evolutionary automatic programming methodology, Grammatical Evolution (GE), is capable of uncovering useful technical trading rules for the US S&P index. In preliminary studies, GE has been applied to this problem using datasets from the FSTE 100, ISEQ, DAX, and the Nikkei, the results showing much promise when compared to a benchmark buy-and-hold strategy [13] [14] [12]. We extend this research by applying the approach to a new market, through the use of a new trading model, and by examining the use of various replacement strategies for the evolutionary search engine.

The paper is organised as follows. Section 2 discusses the background to the technical indicators utilised in this study. Section 3 describes the evolutionary algorithm adopted, Grammatical Evolution [18] [16] [15]. Section 4 outlines the trading methodologies and data and function sets used. The results of the study are provided followed by a discussion of same and finally a number of conclusions are derived.

1.1 Technical Analysis

A market index is comprised of a weighted average measure of the price of individual shares which make up that market. The index value represents an aggregation of the balance of supply and demand for these shares. Market traders,

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 165–170, 2002.

known as technical analysts, believe that prices move in trends and that price patterns repeat themselves [11]. If we accept this premise, that rules, although not necessarily static rules, underlie price behaviour, it follows that trading decisions could be enhanced useing an appropriate rule induction methodology such as Grammatical Evolution (GE). Although controversy exists amongst financial theorists regarding the veracity of the claim of technical analysts, recent evidence has suggested that it may indeed be possible to uncover patterns of predictability in price behaviour. Brock, Lakonishok and LeBaron [2] found that simple technical trading rules had predictive power and suggested that the conclusions of earlier studies that were "premature". Other studies which indicated that there may be predictable patterns in share price movements include those which suggest that markets do not always impound new information instantaneously [8] [4], that stock markets can overreact as a result of excessive investor optimism or pessimism [7], that returns on the market are related to the day of the week [5] or the month of the year [6]. The continued existence of large technical analysis departments in international finance houses is consistent with the hypothesis that technical analysis has proven empirically useful.

1.2 Technical Indicators

The development of trading rules based on current and historic market price information has a long history [3]. The process entails the selection of one or more technical indicators and the development of a trading system based on these indicators. These indicators are formed from various combinations of current and historic price information. Although there are potentially an infinite number of such indicators, the financial literature suggests that certain indicators are widely used by investors [2,11,17].

Four groupings of indicators are given prominence in prior literature: Moving average indicators, Momentum indicators, Trading range indicators, and Oscillators. Given the large search space, an evolutionary automatic programming methodology has promise to determine both a good quality combination of, and relevant parameters for, trading rules drawn from individual technical indicators. We intend to use each of these groupings as our model is developed, but in this investigation we have limited our attention to moving average indicators.

A description of the evolutionary automatic programming system, GE, used to evolve trading rules can be found in [18,16,15].

2 Problem Domain and Experimental Approach

We describe an approach to evolving trading rules using GE for the S&P index using daily data from the period 01/01/1991 to 01/10/1997. The training data set was comprised of the first 440 trading days of the data set. The remaining data was divided into six hold out samples totaling 2190 trading days. The division of the hold out period into six segments was undertaken to allow comparison of the out-of-sample results across different market conditions in order

to assess the stability and degradation characteristics of the developed model's predictions. The extensive hold out sample period helps reduce the possibility of training data overfit. The rules evolved by GE are used to generate one of three signals for each day of the training or test periods. The possible signals are *Buy*, *Sell*, or *Do Nothing*. Permitting the model to output a Do Nothing signal reduces the hard threshold problem associated with production of a binary output. This issue has not been considered in a number of prior studies.

2.1 The Trading Methodology

In the trading model developed the rules evolved by GE generate a signal, which can result in one of three actions to be taken by the model Buy, Sell or Do Nothing. The model will take variable sized positions based on the strength of the signal.

Signals received from GE will oscillate around a pivot point of zero. Signals greater than zero constitute a Buy signal, initially, the first Buy signal received will be recorded as the maximum signal and the maximum amount will be invested, $1,000 (arbitrary). All subsequent investments with lesser signals will be a percentage of the maximum amount in the same ratio as their signal is to that of previous strongest received signal. When a signal is received that is stronger than the previous strongest, it is recorded and the maximum amount again invested. It should also be stated that if the sum to be invested is greater than the cash available, the model will invest the cash available less transaction costs. Upon receipt of a Sell signal, less than zero, all positions are closed.

The total return in this model is a combination of its generated return from market activity and interest gained from the risk free cash position. In this case the interest rates used were the actual American Deposit Interest rates for the same period as the S&P data set, affording an accurate reflection in the cash position.

Transaction costs in this model are based on the cost structure used by online trading houses today, where flat fees are incurred for the opening and closing of positions, $10 fees were charged upon entry and exit. When the model receives weak signals smaller sums are invested. When the transaction costs represent a certain percentage of the sum to be invested, the investment becomes unfeasible. Therefore the model is made aware of the transaction costs, when the entry and exit costs arise to 20% (arbitrary) of the sum to be invested the model will not take the position. This represents the Do Nothing stance, the model will hold all current positions.

In selecting a fitness measure for the model pure return over the training period was selected. A key reason for this was that risk should already be factored into the trading strategy.

3 Results

A preliminary set of experiments were conducted using a number of replacement strategies, varying from 10% up to 100% replacement. Mean's for the best

Table 1. The mean of the best fitness values for both the in-sample and out-of-sample data for the five different replacement strategies compared against the market return over the same periods

	10%	**25%**	**50%**	**75%**	**100%**	**Market**
In-Sample Mean best fitness values	10.065	11.348	11.119	10.351	11.708	8.607
In-Sample Mean average fitness values	7.6546	8.1914	6.7905	5.3267	3.1338	8.607

Table 2. The performance for the best evolved trading ruleset on both the in-sample and out-of-sample data for the five different replacement strategies compared against the market return over the same periods

	10%	**25%**	**50%**	**75%**	**100%**	**Market**
In-Sample Best fitness values	13.052	14.066	15.086	16.086	15.123	8.607
Out-of-Sample Best fitness values	10.042	13.402	11.595	11.109	9.571	70.014

Table 3. Results for the best evolved ruleset from the trading methodology under investigation and the market return benchmark over the out-of-sample data

Model	**Train (1991)**	**1992**	**1993**	**1994**	**1995**	**1996**	**1997**	**Total**
New	14.066%	2%	1%	-4.7%	7.333%	2.07%	5.7%	**13.403%**
Market	8.607%	-1.12%	6.749%	-1.695%	25.5%	16.43%	24.15%	**70.014%**

and average fitness values over 30 runs for both the in-sample data can be seen in Table 1. Statistical analysis of the mean best fitness values using a t-test and bootstrap t-test show that 100% strategy yields the greatest average return followed by the 25% and the 50%. However, the analysis also identifies these three strategies as being statistically equivalent. Analysis of the mean average fitness values shows that the 25% strategy yielded the greatest average return followed by the 10% strategy, however, these performances were deemed statistically equivalent.

The performance of the best evolved individual on both the in and out of sample data can be seen in Table 2.

Table 3 shows the results over the training period and each of the test periods examined for the two models and the market return benchmark. It can be seen that the model never fully exploited the potential for returns that the market held over the latter test periods where the market is in a strong upward trend. The inability of the evolved rules to take maximum benefits from the market may be due in part to the training period used which reflects a relatively diverse period where the rules would benefit from a more conservative trading strategy. Further experiments on more datasets will be required in order to ascertain the true significance of the results presented here.

3.1 Constant Evolution

In this investigation the grammar used in the experiments used a novel method for constant evolution through Digit Concatenation, which differs from the previous approaches adopted with GE where constants were allowed to evolve through expressions. A comparison between the two methods of constant evolution was undertaken through the use of two distinct grammars, extracts of which are shown below, however, no statistical difference (using a t-test and bootstrap t-test) in terms of best fitness performance over 30 independent runs was observed for the two approaches. Mean of the average fitness values however, showed that the concatenation approach is statistically superior. It was also noted that the digit concatenation method for evolving constants allowed the models to evolve large numbers more quickly. These findings suggest that the concatenation method for constant evolution may prove to be a useful tool under a different problem domain. A third grammar was analysed that combined the two approaches, and again, in this case there was no significant difference in performance in terms of mean best fitness, however, on the mean average fitness this approach is shown to be inferior to the pure concatenation approach.

Expression Grammar

```
expressions: expressions op exp | exp
exp: mAvg( numbers ) | numbers
op: - | + | * | /
numbers: numbers op number | number
number: 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Digit Concatenation Grammar

```
expressions: expressions op exp | exp
exp: mAvg( numbers ) op mAvg ( numbers )
op: - | + | * | /
numbers: numbers number | number
number: 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Constant concatenation plus expressions grammar

```
expressions: expressions op exp | exp
exp: mAvg( numbers ) | numbers
op: - | + | * | /
numbers: numbers op numbers | numbers number | number
number: 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

4 Conclusions and Future Work

GE was shown to successfully evolve trading rules that made gains over a 6 year out-of-sample dataset, however in comparison to the benchmark buy-and-hold strategy the full potential of the market was not exploited due to overfitting on the training data. Two methods for constant evolution were examined where the novel digit concatenation approach was observed to yield larger values with more ease than the traditional approach adopted in GE, however, on the problem domain examined here this lead to insignificant performance gains.

There is notable scope for further research utilising GE in this problem domain. Our preliminary methodology has included a number of simplifications, for example, we only considered moving averages, a primitive technical indicator. The incorporation of additional technical indicators may further improve the performance of our approach.

References

1. Allen, F., Karjalainen, R. (1999) Using genetic algorithms to find technical trading rules. *Journal of Financial Economics*, **51**, pp. 245–271, 1999.
2. Brock, W., Lakonishok, J. and LeBaron B. (1992). 'Simple Technical Trading Rules and the Stochastic Properties of Stock Returns', Journal of Finance, 47(5):1731–1764.
3. Brown, S., Goetzmann W. and Kumar A. (1998). 'The Dow Theory: William Peter Hamilton's Track Record Reconsidered', Journal of Finance, 53(4):1311–1333.
4. Chan, L.K.C., Jegadeesh, N. and Lakonishok, J. (1996). 'Momentum strategies', Journal of Finance, Vol. 51, No. 5, pp. 1681–1714.
5. Cross, F. (1973). 'The Behaviour of Stock prices on Friday and Monday', Financial Analysts' Journal, Vol. 29(6), pp. 67–74.
6. DeBondt, W. and Thaler, R. (1987). 'Further Evidence on Investor Overreaction and Stock Market Seasonality', Journal of Finance, Vol. 42(3):557–581.
7. Dissanaike, G. (1997). 'Do stock market investors overreact?', Journal of Business Finance & Accounting (UK), Vol. 24, No. 1, pp. 27–50.
8. Hong, H., Lim, T. and Stein, J. (1999). 'Bad News Travels Slowly: Size, Analyst Coverage and the Profitability of Momentum Strategies', Research Paper No. 1490, Graduate School of Business, Stanford University.
9. Iba H. and Nikolaev N. (2000). 'Genetic Programming Polynomial Models of Financial Data Series', In *Proc. of CEC 2000*, pp. 1459–1466, IEEE Press.
10. Koza, J. (1992). *Genetic Programming*. MIT Press.
11. Murphy, John J. (1999). Technical Analysis of the Financial Markets, New York: New York Institute of Finance.
12. O'Neill M, Brabazon A., Ryan C. (2002). Forecasting Market Indices using Evolutionary Automatic Programming: A case study. Genetic Algorithms and Genetic Programming in Economics and Finance, Kluwer Academic Publishers 2002, in print.
13. O'Neill M., Brabazon A., Ryan C., Collins J.J. (2001). Evolving Market Index Trading Rules using Grammatical Evolution. In Applications of Evolutionary Computing, Proc. of EvoWorkshops 2001. LNCS 2037, pp. 343–352.
14. O'Neill M., Brabazon A., Ryan C., Collins J.J. (2001). Developing a Market Timing System using Grammatical Evolution. In Proc. of GECCO 2001, pp. 1375–1381.
15. O'Neill M. (2001). Automatic Programming in an Arbitrary Language: Evolving Programs with Grammatical Evolution. PhD thesis, University Of Limerick, 2001.
16. O'Neill M., Ryan C. (2001) Grammatical Evolution. *IEEE Trans. Evolutionary Computation*. 2001.
17. Pring, M. (1991). Technical analysis explained: the successful investor's guide to spotting investment trends and turning points, New York: McGraw-Hill Inc.
18. Ryan C., Collins J.J., O'Neill M. (1998). Grammatical Evolution: Evolving Programs for an Arbitrary Language. *Lecture Notes in Computer Science 1391, Proceedings of the First European Workshop on Genetic Programming*, pp. 83–95. Springer-Verlag.

An Interactive Story Engine

Chris Fairclough and Pádraig Cunningham

ML group, CS Dept., Trinity College Dublin
{Chris.Fairclough,Padraig.Cunningham}@cs.tcd.ie

Abstract. This document describes an approach to story description that allows an agent to dynamically control the plot of a story in a computer game. The way of describing stories that is used is based on Vladimir Propp's work on the structure of folktales. This work is modified for use in a case-based planning and constraint satisfaction system that controls the character agents in a story world and tries to make them follow a coherent plot and react consistently with the player's actions.

1 Introduction

Stories in computer games have so far been strictly pre-authored, with any interactivity built in only as branching plot trees. This is an approach which does not recognise a fundamental property of stories that has been forgotten since oral storytelling and theatre have lost their status as a popular source of folktales. Stories are fundamentally evolving, malleable, adaptive things that react to their audience and teller. A systematic description of stories that does not take this into account will not facilitate interactivity when implemented. However, Vladimir Propp's description in [9] is an analysis of many stories. Propp finds the common properties of different versions of these and proposes a set of primitives and rules that define a flexible story. The primitives are called character functions and are independent of the characters that perform them. This means that although the plot varies if different characters perform the functions at different times, the underlying story structure remains constant provided the same set of functions occur.

The interactive story engine described in this paper is built on Propp's ideas, and aims to facilitate an immersive story-based computer game while allowing the player to influence the plot to a considerable degree. The main software components are: the game world, the Non-Player Character (NPC) agents, and a story director agent. The game world is a simple OpenGL 3D environment with different locations, and usable objects. The NPC agents are simple autonomous reactive agents that can also be assigned goals by the story director agent. The characters in the game world are simple agents that are capable of limited autonomous behaviours, and react to a player if the player enters the game, but the higher level control of the characters is done by a story director (SD) agent that monitors all the character s actions, including the hero (the player).

This paper first provides a background of research in computer-mediated storytelling and assesses a range of different approaches. Then the game world and

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 171–176, 2002.

story representation schemes used in this system are described, with implementation details. The role of Propp's 'character functions' in the system, and the use of case based planning and constraint satisfaction techniques is then elaborated. Some story elements must be pre-authored by a user, and these are identified in the next section, before some conclusions are drawn in the last section.

2 Previous Work

Research on interactive stories has its roots in the arts of oral storytelling and theatre, and ideas from these fields have been incorporated into new forms of storytelling using the computer as a medium. Amongst the earliest projects based on the use of a computer storyteller was 'Tale Spin' by James Meehan [1] which composes stories by giving each character some goals to achieve and forming plans for each of the characters.

2.1 Research in Story Theory

The theory and history of stories is a field that seems to be separated from computer science by a wide gulf. However, Propp's 'Morphology of the Folktale' [9] is a detailed study of the structure of folktales, whose relevance to interactive storytelling is greater than it first appears. The nature of computer storytelling can be likened to the old tradition of oral storytelling in that the narrator reacts in real time to the audience, adjusting the story appropriately. Therefore an analysis of folklore, like Propp's, should be a better starting point for computer storytelling than one of purely linear stories, such as novels or movies.

Turning to other research, Polti's classification of plots [12] has not been as fruitful as Propp's because it does not help deconstruct plots to primitives. There is also work on how the use of dramatic arcs can be applied to computer mediated stories [7]. The problem here is that these arcs, which depict the rise of tension in the spectator while observing a story, are an *effect* of the story, not a *cause* of it.

2.2 Directions in Computer Storytelling

Since an early interest in the topic in the 70's, research in computer storytelling was non-existent until relatively recently. Carnegie Mellon University was a hotspot of interest in the field in the early 90's, spearheaded by Joseph Bates [10,11]. The OZ project [2] at CMU consists of a group of researchers interested in diverse aspects of the field. Some of these researchers have papers posted on Micael Mateas' website [3] which cover the field of Narrative Intelligence. NI, loosely, is an analysis of the human talent required for structuring the world around us into narratives or stories. Some, such as Umberto Eco [13] think that such structuring is a fundamental activity of human intelligence. This ability can be modeled by a computer and used for, among other things, interactive storytelling.

There are three main approaches to computer storytelling [6]: character based, plot based, and author based. The first [5] is an approach where the characters are seen as the most important ingredient of a story. Therefore, it is argued, if sufficiently broad and believable autonomous characters are placed before the player, then a story will emerge from the player's interactions. The second approach [6] is where a plot managment system is used to directly control the actions of characters, leaving only the lowest level behaviours to the agents. Finally, the last approach tries to model an author's thought processes in coming up with a story [8].

3 Game and Story Mechanics for the Engine

A plot is defined both as a series of character functions and a series of complication-resolution event pairs and that can be nested. Character functions are seen as the means of creating complication or resolution events. Other events can happen in the world that are not directly relevant to the plot, but link story events. Story events are broadly categorized into complication creation events or complication resolution events. A complication is created when a character performs some function which alters the situation of the hero, whether he knows of it or not, for example putting a challenge before him.

The game world is made up of characters, locations, and objects. Locations are navigated by characters using special objects called portals. There are also special objects called action objects, that enable characters to perform actions on other characters or objects. An example would be a sword, which injures other characters and generates a negative change in one character's rating for the other.

The characters are the agents that carry out the story plan. A character is defined by an internal state, which is initialised by the story author. The state of a character is changed due to interactions with other characters. It also influences his actions and reactions. Some of a character's actions are called character functions; these drive the plot along, and are initiated by the SD agent by giving the character a certain goal.

There are four kinds of character behaviour: (1) low level – collision detection which steers away from nearby objects and characters as they get too close; (2) primitive social interactions – if a character meets another that they have not met before, they will introduce themselves; (3) idle behaviours – an example is the 'potter' behaviour which makes a character hang around beside a specified object or character; (4) targeted behaviour – the SD can give a character goals, and it will look for the object of the goal.

Two characters are defined as the hero and villain before the story starts. Other characters may be defined as helpers to either party, but their loyalties may not be clear to the player at the beginning of the story.

4 Character Functions in Cases

Character functions are those character actions that affect the development of the story [9]. There are many different types of functions which include the helping or hindering of the hero, and the movement function which transports the hero to another part of the story world. Cases in the case based planning system are made up of scripts which consist of lists of these character functions. One common misunderstanding and criticism of Propps's work comes from the assumption that he intends these functions as the most basic primitives of stories. Propp refers to the functions as *species*, each with any number of *genera*, of which he gives some examples. Each function iteration within cases will be given using a scripting language where this kind of simple syntax will be used:

```
Villainy(steal(familymember));
Test(quest(desiredobject));
```

- The Villainy function is achieved by kidnapping some character close to the hero. The reference to family in Propp's analysis will not be seen as literal for this implementation.
- The Quest is a basic element of stories, and has strong correllations in computer games.

Within each case, certain functions may be repeated. For longer stories, a number of cases can occur back to back, or simultaneously with different heroes. A comrade of the hero could have his own hero's journey side by side with the hero. This property could lend itself to a multiplayer version of the story engine.

The user influences the plot by moving the hero character around, and performing various actions on the other characters. If a certain reaction is required of the hero in order to move the plot forward, functions can be repeated to exert some influence on the player, e.g. the Villainy function could be repeated until the hero decides to counteract the villain.

Different cases have different genera of functions, and the presence of a complication function of a certain genus implies a corresponding resolution function from the same case. For this to work, the complication and resolution functions that go together must be annotated as such within the cases, or else each case must only have one complication/resolution pair.

4.1 Casting as Constraint Satisfaction

The selection of characters to perform the functions will be termed *casting*. This activity also includes the determination of objects for use as objects of quests, and character subjects of villainy. This task is done with a constraint satisfaction system. In the kidnapping of a family member, for example, the casting system will be given the constraint that a character must have a close relationship to the hero character, as established with the social modelling. Each story world could have different definitions for 'close relationship'. This provokes the question of

how closely the game world is integrated with the story director. Ideally the director is loosely coupled, making some basic information requests of the game world when required.

The 'spheres of action' that Propp names are Hero, Villain, Helper, Donor, Princess, Mediator and False Hero. These will be used as *roles* in the story system. Of these, the hero will be a constant, cast as the player and the rest are cast dynamically. In most stories, the villain is also a constant. Each role has a number of constraints that must be fulfilled as much as possible in the choice of character.

5 Authored and Non-authored Story Elements in an Interactive Story Engine

For use in a computer game, the engine must allow the author of a story to define certain elements of the story, and then it must extrapolate the rest of the story elements from this basic description.

5.1 Pre-authored Elements

Apart from the aesthetic things like creating the graphical story world and character models, the author must set up an identity for each character which includes loyalties (toward the hero or villain, and to what degree?), links to world objects that belong to the character, and links to characters in the world that this character knows.

There are techniques that are widely used in commercial games for storytelling purposes, such as cut scenes which show how the plot progresses after each scene of the game's action. These could be incorporated into this engine, but only at points in a plot where all complications are resolved, and the next scene of action starts fresh. In this way, a plot could be authored as a series of episodes whose detail is controlled by the SD.

5.2 Non-authored Elements

The SD views the events in the story world from each character's perspective, and controls how, or if at all, each character will react to things it witnesses. Witnessed events could cause a character to form plans of its own. At this stage of development, it is thought that most of these plans will be contained in the SD agent, while simple tasks, such as the task of getting to a certain location, will be farmed out to the character class.

An important element of the story engine is the means of transferring knowledge that concerns story events between characters; a 'gossip' engine. This could be done at the character level, but would mean each character building up a large, and mostly redundant, store of information about the other characters. The gossiping could be handled by the SD which notes each event in the story, and also which characters witnessed each event.

6 Conclusions

The Architecture described here is in the process of being implemented, and a demo movie can be found at
`http://www.cs.tcd.ie/Chris.Fairclough/peanut1.avi`.
This movie shows a simple quest story being 'acted out' by the characters, with the player as a passive participant in the action, simply moving around the environment without performing any actions. There is no case based planning system yet, as the constraint satisfaction system is being more fully explored as a means to allow interactivity in the story. The authoring of cases is a task of analysing existing popular stories, and sometimes shoehorning them into Propp's structures. Propp himself provides a list of Russian folklore plots as character functions, so these will be used in the case base.

Although this is a means of integrating story into a game at the game mechanics level, rather than as a motivation for gameplay, computer games usually have a challenge to them, and this story engine is intended to be used as a component of a full game engine. There are many potential additions that can be made, such as a multiplayer version of the engine over a network, which will allow a player to assume different story roles, and to change the characters they are controlling while playing.

References

1. Meehan, J.: The metanovel: writing stories on computer. University microfilms international, 1977.
2. `http://www.cs.cmu.edu/afs/cs.cmu.edu/project/oz/web/`: The OZ project at Carnegie Mellon University, Pittsburgh.
3. `http://www.cs.cmu.edu/~michaelm/nidocs/`: A site with many Narrative Intelligence papers.
4. Michael Mateas, Phoebe Sengers: Narrative Intelligence, 1999.
5. Rizzo P., Veloso M.V., Miceli M., Cesta A: Goal-based Personalities and Social Behaviors in Believable Agents.
6. Michael Mateas, Andrew Stern: Towards Integrating Plot and Character for Interactive Drama, 1999.
7. Michael Mateas, Andrew Stern: Towards building a fully-realised interactive drama.
   `http://www.stg.brown.edu/conferences/DAC/subs_in/Mateas.html`, 2001.
8. Kahn M.: Creation of Computer Animation from story descriptions. PhD thesis, AI lab, MIT, 1979.
9. Vladimir Propp: Morphology of the Folktale, 1968.
10. Sean Smith and Joseph Bates: Towards a Theory of Narrative for Interactive Fiction, CMU-CS-89-121, 1989.
11. W. Scott Reilly and Joseph Bates: Building Emotional Agents, CMU-CS-92-143, 1992.
12. `http://www.geocities.com/SiliconValley/Bay/2535/polti.html`.
13. Umberto Eco: Six Walks in the Fictional Woods Harvard University Press, 1994.

Coherence, Explanation, and Bayesian Networks

David H. Glass

Faculty of Informatics, University of Ulster, Newtownabbey, BT37 0QB, UK
dh.glass@ulster.ac.uk

Abstract. This paper discusses the relevance of coherence to deciding between competing explanations. It provides a basic definition of coherence in probabilistic terms, which yields a coherence measure and can easily be extended from the coherence of two beliefs to the coherence of n beliefs. Using this definition, the coherence of a set of beliefs can be obtained by making simple extensions to a Bayesian network. The basic definition suggests a strategy for revising beliefs since a decision to reject a belief can be based on maximising the coherence of the remaining beliefs. It is also argued that coherence can provide a suitable approach for inference to the best explanation.

1 Introduction

In a wide range of circumstances people try to make sense of certain events by coming up with an explanation as to how the events could have occurred. The process of generating explanations often involves causal reasoning from the observed effects to their causes. However, in addition to producing suitable explanatory hypotheses the question of deciding between competing explanations naturally arises. Such reasoning processes occur in medical diagnosis, fault diagnosis, legal cases and scientific reasoning to name just a few. This approach to reasoning is often referred to as inference to the best explanation or abduction and has been a major topic in epistemology [1-3] as well as in artificial intelligence [4-6]. Intuitively it seems that the coherence of an explanation may be important in this context since, for example, we would not take seriously an explanation which we know to be incoherent. The notion of coherence is particularly prominent in the work of Thagard [5,6] who uses coherence to determine the best of a number of competing explanations.

Another approach to causal reasoning has been to use probability theory which has been very successful within artificial intelligence [7,8] and is the approach often adopted for reasoning under uncertainty. The idea in this paper is to bring diverse approaches together by defining coherence in terms of probability. Recent work along similar lines has been carried out by Bovens and Olsson [9] and Bovens and Hartmann [10] who address philosophical disputes over coherentism and use Bayesian networks to study coherence and belief expansion. The novel aspect of the current paper is to give a different, but very simple, definition of a coherence measure for a set of beliefs which can be obtained from a Bayesian network in a straightforward manner. In the rest of the paper, Sect. 2 discusses the concept of coherence and presents a definition for the coherence between two beliefs. Section 3 looks at a very

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 177-182, 2002.

powerful approach where useful information about coherence can be extracted from Bayesian networks. Section 4 considers how the definition of coherence may be used in inference to the best explanation. Finally, Section 5 discusses potential applications and presents conclusions.

2 Defining Coherence

Despite considerable discussion on the subject of coherence no agreed definition is forthcoming. However, Bovens and Olsson [9] note that there us agreement that a coherent set of beliefs should be a set whose members provide mutual support for each other. This suggests that a definition of coherence in terms of conditional probability may be possible. In another paper [10] such a coherence measure is defined, but a possible problem with the definition is that coherence depends on the reliability of the sources that provide us with the beliefs. Instead, it seems that the coherence of beliefs A and B should depend only on the relation of A to B and not on how likely A or B are in the first place.

In this paper a very basic definition of coherence of two beliefs A and B, denoted C(A,B), is considered. C(A,B) is defined as

$$C(A,B) = \frac{P(A,B)}{P(A \vee B)} \ , \tag{1}$$

provided $P(A \vee B) \neq 0$. According to this definition coherence is related to, but not identified with, joint probability. It is not just the extent of agreement between the two beliefs that determines their coherence, but the extent of their disagreement as well. To put it another way, it is their 'degree of overlap' that is important (see Fig. 1). This seems to fit in with our intuitive notion that beliefs can be extremely coherent even if they are improbable.

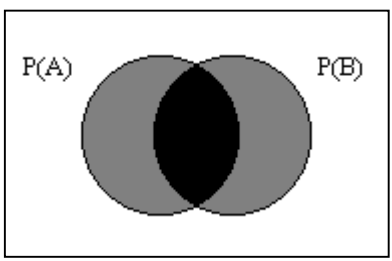


Fig. 1. A diagram to represent coherence. The coherence of A and B, C(A,B), is the ratio of the area shaded black to the total area contained within the circles

In the case where P(A,B) = 0 then C(A,B) = 0, whereas when A entails B and vice-versa then C(A,B) = 1. Thus, as with probability, coherence yields a measure on the interval [0,1],

$$0 \leq C(A,B) \leq 1 \ . \tag{2}$$

By using Bayes' theorem and assuming that P(A,B) ≠ 0, (1) can be rewritten as,

$$C(A,B) = \frac{P(A \mid B)P(B)}{P(A) + P(B) - P(A \mid B)P(B)} = \left[\frac{1}{P(B \mid A)} + \frac{1}{P(A \mid B)} - 1\right]^{-1}. \quad (3)$$

Note that by writing the expression for coherence in this way it can be expressed in terms of conditional probabilities.

A possible problem seems to arise when more than two beliefs are taken into account. Consider two scenarios: in scenario one, P(A,B) = P(B,C) = P(C,A) = 0 and so P(A,B,C) = 0, i.e. there is no overlap between any pair of beliefs; in scenario two, P(A,B), P(B,C) and P(C,A) are all non-zero, but P(A,B,C) = 0. In both cases C(A,B,C) = 0, yet perhaps we would want to say that the beliefs in scenario two are more coherent. However, in scenario two the beliefs A, B and C are inconsistent since they cannot all be true simultaneously and so a coherence of zero seems reasonable. Perhaps the difference lies in the fact that in scenario two a degree of coherence can be attained by rejecting one of the beliefs, C say. In contrast, rejecting one of the beliefs in scenario one makes no difference since the coherence remains at zero.

The above discussion suggests a strategy for revising beliefs. In a case such as scenario two, where the coherence is zero, a decision to reject a belief can be based on maximising the coherence of the remaining beliefs. Alternatively, suppose that someone holds beliefs A and B, but then comes across very strong evidence for C. In this case the person should reject A or B on the basis of retaining as coherent a set of beliefs as possible.

3 Coherence and Bayesian Networks

The current definition of coherence can readily be extended to more than two beliefs. The coherence of n beliefs $A_1, A_2, \ldots, A_n$ can be written as

$$C(A_1, A_2, \ldots, A_n) = \frac{P(A_1, A_2, \ldots, A_n)}{P(A_1 \vee A_2 \vee \ldots \vee A_n)} = \left[\frac{1 - P(A_n / A_1 \vee A_2 \vee \ldots \vee A_{n-1})}{C(A_1, A_2, \ldots, A_{n-1})P(A_n \mid A_1, \ldots, A_{n-1})} + \frac{1}{P(A_1, \ldots, A_{n-1} \mid A_n)}\right]^{-1} \quad (4)$$

If the notion of coherence is to be used for inference to the best explanation then it would certainly be beneficial to have an efficient means of computing the coherence measure. Furthermore, given the success and widespread use of Bayesian networks it would seem to be advantageous to incorporate coherence into these networks. Fortunately, this turns out to be fairly straightforward. The implementation is very similar

to that of Bovens and Hartmann [10] and requires two extra nodes to be added to the Bayesian network: one for the probability of the conjunction of the beliefs in the network and one for the disjunction. There will also be an arc from the node for each belief to each of these additional nodes (see Fig. 2).

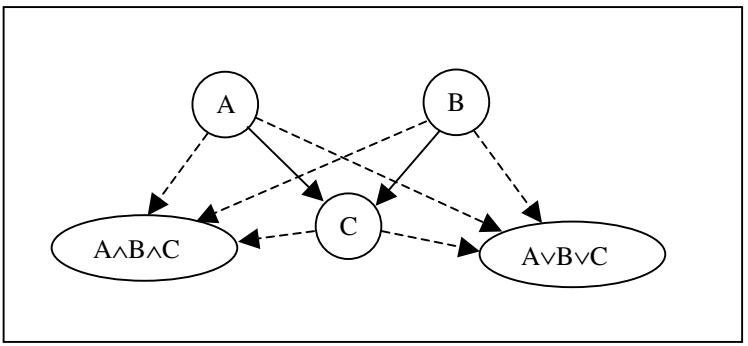


Fig. 2. A Bayesian network with just three beliefs A, B and C which has been extended by two nodes to calculate the coherence. The dashed lines are also part of the extended network.

The conditional probabilities for the node A∧B∧C are

$$\begin{aligned} &P(A \wedge B \wedge C \mid A, B, C) = 1, \\ &P(A \wedge B \wedge C \mid A, B, \neg C) = 0, \\ &\ldots \\ &P(A \wedge B \wedge C \mid \neg A, \neg B, C) = 0, \\ &P(A \wedge B \wedge C \mid \neg A, \neg B, \neg C) = 0. \end{aligned} \tag{5}$$

and for A∨B∨C are

$$\begin{aligned} &P(A \vee B \vee C \mid A, B, C) = 1, \\ &P(A \vee B \vee C \mid A, B, \neg C) = 1, \\ &\ldots \\ &P(A \vee B \vee C \mid \neg A, \neg B, C) = 1, \\ &P(A \vee B \vee C \mid \neg A, \neg B, \neg C) = 0. \end{aligned} \tag{6}$$

The ratio of the probability of node A∧B∧C to that of A∨B∨C gives the coherence of the three beliefs. This simple way of obtaining the coherence measure from a Bayesian network illustrates that for any problem to which the Bayesian network approach can be applied it is a straightforward matter to calculate the coherence for different combinations of the beliefs as represented by the nodes.

4 Inference to the Best Explanation

It may seem that probability provides a straightforward way of determining which of two (or more) hypotheses gives the best explanation for a set of data. To see, however, that things are not straightforward consider two theories/hypotheses T_1 and T_2 which are mutually exclusive so that $P(T_1|T_2) = 0$ and a piece of evidence E for which the theories offer competing explanations. One suggestion might be that T_1 is the better explanation and so should be chosen if

$$P(E \mid T_1) > P(E \mid T_2), \tag{7}$$

but it could be that, even though $P(E|T_1)$ is high (in fact it could be that $P(E|T_1) = 1$), $P(E|\neg T_1)$ is also quite high. Perhaps an improvement on (7) would be to consider likelihood ratios so that T_1 is chosen if

$$\frac{P(E \mid T_1)}{P(E \mid \neg T_1)} > \frac{P(E \mid T_2)}{P(E \mid \neg T_2)}. \tag{8}$$

One difficulty with this approach is that it may give an unfair weighting to $P(E|\neg T_i)$. Consider the most extreme case when $P(E|\neg T_1) = P(E|\neg T_2) = 0$. In this case there is no way to decide between T_1 and T_2, yet intuitively we would think that T_1 was a better explanation if (7) held.

Furthermore, if the prior probability of T_1 is low then the posterior probability of T_1 (i.e. $P(T_1|E)$) may be much smaller than $P(T_2|E)$ despite (7) holding. This suggests instead that T_1 should be accepted if

$$P(T_1 \mid E) > P(T_2 \mid E). \tag{9}$$

However, this may well mean that T_1 is accepted even in the case where E is rather unlikely given T_1, i.e. T_1 is compatible with both E and its negation. This would be the case if T_1 was rather vague, for example.

An alternative approach is to use the notion of coherence to adjudicate between the two theories. Thus T_1 should be accepted if

$$C(T_1, E) > C(T_2, E). \tag{10}$$

Coherence seems to yield a compromise so that $P(E|T_1)$ and $P(T_1|E)$ are both important in deciding between the theories. Adopting this method means that the best explanation of a piece of evidence is taken to be the theory/hypothesis that best coheres with the evidence.

Coherence can also be used to shed some light on the simplicity and testability of hypotheses. In the case of simplicity this is due to the fact that the coherence of n beliefs is less than or equal to the coherence of a subset of those beliefs. In the case of testability the coherence of a set of beliefs can tell us something about the impact of verifying or falsifying one of the beliefs. See Glass [11] for further discussion.

5 Conclusions

This paper has presented a very simple definition of coherence in probabilistic terms. This definition may be rather limited and better alternatives may be available, but it does provide a starting point for thinking about the role of coherence in artificial intelligence. An important aspect of coherence as defined in this paper is that it can be incorporated into a Bayesian network. This opens up a wide range of potential applications. Applications may also arise from the fact that in many cases it is desirable to have hypotheses which are amenable to testing and this is particularly true in medical diagnosis. A recent article by McSherry [12] discusses how tests should be selected in sequential diagnosis. A similar strategy should be possible based on the current approach except that the target diagnosis would be the one that is the most coherent with the evidence rather than the most probable.

Further work needs to be carried out in applying these ideas to particular cases and also to focus in more detail on a number of issues. For example, the idea presented in Sect. 4 that coherence can be used to determine the best explanation needs to be compared in detail with alternative approaches adopted within machine learning.

Acknowledgements. The author would like to thank Prof. David Bell for helpful discussions.

References

1. Harman, G.: Change in view: Principles of reasoning. Cambridge, MA: MIT Press (1986).
2. Lipton, P.: Inference to the Best Explanation. London: Routledge (1991).
3 Lycan, W.: Judgement and Justification. Cambridge: Cambridge University Press (1988).
4. Josephson, J.R.: Josephson S.G.:Abductive Inference: Computation, philosophy, technology. Cambridge: Cambridge University Press (1994).
5. Thagard, P.: Explanatory Coherence. *Behavioural and Brain Sciences* **12** 435-467 (1989).
6. Thagard, P.: Probabilistic Networks and Explanatory Coherence. *Cognitive Science Quarterly* **1** 93-116 (2000).
7. Neapolitan, R.: Probabilistic Reasoning in Expert Systems, New York: John Wiley (1990).
8. Pearl, J.: Probabilistic Reasoning in Intelligent Systems, San Mateo: Morgan Kaufman (1988).
9. Bovens, L., Olsson, E.J.: Coherentism, Reliability and Bayesian Networks, *Mind* **109** 685-719 (2000).
10. Bovens, L., Hartmann, S.: Coherence, Belief Expansion and Bayesian Networks, *Proceedings of the 8th International Workshop on Non-Monotonic Reasoning*, NMR'2000 Breckenridge, Colorado, USA, April 9-11, (2000).
11. Glass, D.H.: A Probabilistic Account of Coherence, *Proceedings of the International Conference on AI*, IC-AI'2002 Las Vegas, Nevada, USA, June 24-27, (2002).
12. McSherry, D.: Sequential Diagnosis in the Independence Bayesian Framework, in D. Bustard, W. Liu, R Sterritt (Eds.) *Soft-Ware 2002:Computing in an Imperfect Word*, 217-231, Springer (2002).

Combining Case-Based Reasoning and Analogical Reasoning in Software Design

Paulo Gomes, Francisco C. Pereira, Nuno Seco, Paulo Paiva, Paulo Carreiro, José L. Ferreira, and Carlos Bento

CISUC – Centro de Informática e Sistemas da Universidade de Coimbra. Departamento de Engenharia Informática, Polo II, Universidade de Coimbra. 3030 Coimbra
{pgomes,camara,zeluis,bento}@dei.uc.pt
{nseco,paiva,carreiro}@student.dei.uc.pt
http://rebuilder.dei.uc.pt

Abstract. Designers use several types of knowledge and reasoning mechanisms during the creation of new artefacts. In order to cope with this cognitive characteristic of design, an intelligent design tool able to help a designer must integrate several reasoning mechanisms and knowledge formats. Case-based reasoning and analogical reasoning are usually considered as two distinct mechanisms, though they are also considered to be in the same cognitive axis, case-based reasoning being in one extreme, and analogy in the other. Both are important reasoning mechanisms in the generation of new designs, but they both reflect different ways of exploring the design space. In this paper we present a way of combining both techniques, showing how it was integrated in an intelligent software design tool. Experimental results are presented and discussed, showing the advantages and limitations of this approach.

1 Motivation and Goals

Design is a complex activity that is generally ill-structured and requires a great amount of different types of knowledge. The design process generates a structural description that complies with a set of functional specifications and constraints. The design outcome is a set of components and relations between them, making the design structure. The design process involves three main types of knowledge about the domain: functional, behavioural and structural. The mapping between these three types of knowledge is central to the design process.

In order to provide a design system with several reasoning mechanisms, we have developed a system that integrates case-based reasoning [1] and analogy reasoning [2]. These two reasoning mechanisms provide support for the inexperienced designer, showing and allowing the reuse of several relevant past designs, and it can also help the more experienced designer to explore new solutions far away from the problem domain.

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 183-189, 2002.

Several related research works have been done, from which we present some of the most relevant ones. The Holographic Reduced Representations approach [3] combines semantic and structural similarity. The semantic similarity is based on the common semantic primitives of objects, while the structural similarity is accessed using the various roles of each object. RADAR developed by Crean and O'Donoghue [4], was created to retrieve semantically distant analogies in a computationally tractable manner, and is based only in structure mapping. Déjà vu [5] is a CBR systems for code generation and reuse using hierarchical CBR. Déjà Vu uses a hierarchical case representation, indexing cases using functional features. Maiden and Sutcliffe [6] research work is centred on the analogy in software specification reuse. Their approach is based on the reasoning with critical problem features, retrieving analogous specifications of the same category.

In our approach we have combined Case-Based Reasoning (CBR) with analogy using the retrieval phase of both reasoning methods as a common point. While CBR centers in semantic similarity, analogy approaches commonly use structural similarity. Another usual cited difference is that the goal of analogy is to establish a mapping between different domains, while CBR maps concepts between the same domain. We use a general ontology to establish the connection between both mechanisms, and specifically object categorization as a mean of retrieving semantic similar concepts for CBR, and distant semantic concepts for analogy. We have develop a software design system named REBUILDER based on this theory. In the next section we describe REBUILDER, showing its architecture, ontology used, CBR module, and analogy module, and then the experimental work is presented.

2 REBUILDER

REBUILDER is a software design CASE tool, with the goal of providing intelligent help for software developers. There are two types of users interacting with the system. The software designers, using REBUILDER as an UML editor and design support tool, and the system administrator with the main task of keeping the knowledge base (KB) updated. In our approach we use one of the most worldwide used software design languages: the Unified Modelling Language, best known as UML [7].

REBUILDER architecture comprises four main modules: UML editor, KB management, KB, and CBR engine. The UML editor is the system front-end for the software designer, comprising the working space for design. The KB management is the interface between the KB and the system administrator. It allows the administrator to add, delete, or change knowledge from the KB.

The KB comprises four sub modules: data type taxonomy, case library, case indexes, and WordNet knowledge. The data type taxonomy provides *is-a* relations between data types used in UML. The case library stores the design cases, each one representing a software design. The case indexes are used for the case retrieval, allowing a more efficient retrieval. WordNet is a lexical reference system [8], used in REBUILDER as a general ontology that categorizes case objects.

The CBR engine performs all the inference work in REBUILDER. There are five sub modules: retrieval, analogy, adaptation, verification, and learning. The retrieval module searches the case library for designs or objects similar to the query. Retrieved designs are presented to the user, allowing the user to reuse these designs (or pieces of them), or they can also suggest new ideas to the designer, helping him to explore the design space. Analogy uses the retrieval module to select the most similar designs from the case library, and then maps them to the query design. The resulting mapping establishes how the knowledge transfer from the old design to the query design is done. The adaptation module can be used to adapt a past design (or part of it) to the query design. The verification module checks the current design for inconsistencies. The learning module acquires new knowledge from the user interaction, or from the system reasoning. This paper focus on the retrieval and analogy modules.

2.1 WordNet

WordNet is used in REBUILDER as a common sense ontology. It uses a differential theory where concept meanings are represented by symbols that enable a theorist to distinguish among them. Symbols are words, and concept meanings are called synsets. A synset is a concept represented by one or more words. If more than one word can be used to represent a synset, then they are called synonyms. The same word can also have more than one different meaning (polysemy).

WordNet is built around the concept of synset. Basically it comprises a list of synsets, and different semantic relations between synsets. The first part is a list of words, each one with a list of synsets that the word represents. The second part, is a set of semantic relations between synsets. In REBUILDER, we use the word synset list and four semantic relation: *is-a*, *part-of*, *substance-of*, and *member-of*. REBUILDER uses synsets for categorization of software objects. Each object has a context synset which represents the object meaning. In order to find the correct synset, REBUILDER uses the object name, and the names of the objects related with it, which define the object context. The object's context synset can then be used for computing object similarity, or it can be used as a case index.

2.2 Case Retrieval

REBUILDER uses a CBR framework to manage the different reasoning modules. Cases are represented in UML. REBUILDER allows the creation and manipulation of all kinds of diagrams, though only class diagrams are used for reasoning purposes. A class diagram comprises three types of objects (packages, classes and interfaces) and different relations between objects (associations, generalizations, dependencies, and realizations).

A case is an UML class diagram, which has a main package named root package (since a package can contain sub packages). REBUILDER can retrieve cases or pieces of cases, depending on the user query. In the first situation the retrieval module returns packages only, while in the second one, it can retrieve classes or interfaces. The retrieval module treats both situations the same way, since it goes to the case library searching for software objects that satisfy the query.

The retrieval algorithm has two distinct phases: first uses the context synsets of the query objects to get *N* objects from the case library. Where *N* is the number of objects to be retrieved. This search is done using the WordNet semantic relations that work like a conceptual graph, and with the case indexes that relate the case objects with WordNet synsets. The second phase ranks the set of retrieved objects using object similarity metrics.

In the first phase the algorithm uses the context synset of the query object as entry points in WordNet graph. Then it gets the objects that are indexed by this synset using the case indexes. Only objects of the same type as the query are retrieved. If the objects found do not reach *N*, then the search is expanded to the neighbour synsets using only the *is-a* relations. Then, the algorithm gets the new set of objects indexed by these synsets. If there is still not enough objects, the system keeps expanding until it reaches the desired number of objects, or there is no where to expand.

The result of the previous phase is a set of *N* objects. The second phase ranks these objects by similarity with the query. Ranking is based on object similarity, and there are three types of object similarities: package similarity, class similarity, and interface similarity.

2.3 Analogical Reasoning

Analogical reasoning is used in REBUILDER to suggest class diagrams to the designer, based on a query diagram. The analogy module has three phases: identifying diagrams candidate for analogy, mapping candidate diagrams, and creation of new diagrams.

Most of the analogies that are found in software design are functional analogies, that is, the analogy mapping is performed using the functional similarity between objects. Since functional similar objects are categorized in the same branch (or near) in the WordNet *is-a* trees, the retrieval algorithm uses these semantic relations to retrieve objects. The analogy module uses the retrieval module for the first phase – Candidate Selection.

The second phase of analogy is the mapping of each candidate to the query diagram, yielding an object list correspondence for each candidate. This phase relies on a relation-based mapping algorithm, which uses the UML relations to establish the object mappings. It starts the mapping selecting a query relation based on an UML heuristic that selects the relation connecting the two most important diagram objects. Then it tries to find a matching relation on the candidate diagram. After it finds a match, it starts the mapping by the neighbour relations, spreading the mapping using the relations. This algorithm maps objects in pairs corresponding to the relation's objects, satisfying the structural constraints defined by the UML diagram relations.

For each set of mappings the analogy module creates a new diagram, which is a copy of the query diagram. Then using the mappings between the query objects and the candidate objects, the algorithm transfers knowledge from the candidate diagram to the new diagram.

3 Experimental Results

The Knowledge Base used comprises a case library with 60 cases, distributed in five domains. Each case comprises a package, with 5 to 20 objects. Each object has up to 20 attributes, and up to 20 methods. Three sets of problems were specified, based on stored cases. Problem set P20, comprises 25 problems, each problem is a case copy with 80% of its objects deleted, attributes and methods are also reduced by 80%. The other sets comprise the same problems but with 50% (P50) and 20% (P80) deletions.

The goal of the experiments is to compare the candidate selection criteria. For each problem we applied the analogy mapping to all cases of the KB, thus serving as a base line for evaluation. The other candidate strategies are: CBR+Analogy using type similarity, CBR+Analogy using structural similarity, and CBR+Analogy using both type similarity and structural similarity. For each problem four runs were made:

Run	Similarity type used	Run	Similarity type used
C1 - Analogy Mapping	Structural	C3 - CBR + Analogy	Structural (1) + Type (0)
C2 – CBR + Analogy	Structural (0.5) + Type (0.5)	C4 - CBR + Analogy	Structural (0) + Type (1)

For each problem run the best five new cases were analysed, based on the mappings performed by the analogy algorithm. The data gathered was: percentage of mappings in the new case considered as correct, number of mappings, distance of mapped objects, depth of the Most Specific Common Abstraction (MSCA) between mapped objects, and computation time.

The results obtained for the first five solutions generated by the analogy module are presented in Table 1. The values presented are the average results of the three problem sets (P20, P50 and P80). These results show a clear trade-off between computation time and quality of results. Analogy by its own generates better solutions, which can be seen by the percentage of correct mappings. This reflects also in the shorter semantic distance between objects, and the greater depth of the MSCA. Configurations which use case retrieval are faster then C1.

Table 1. Experimental results obtained for the first five solutions presented by REBUILDER, and for the three problem sets (average of values for P20, P50 and P80)

	C1	C2	C3	C4
Average of Computation Time (milliseconds)	2903	2497	2479	2279
Average of Percentage of Correct Mappings (%)	64	52	53	45
Average of Number of Mappings	3.15	3.02	3.02	2.97
Average Distance of Mapped Objects (is-a links)	2.94	3.48	3.49	3.85
Average Depth of MSCA (is-a links)	3.56	3.32	3.31	3.11
Average Number of Mappings x Correct Percentage	2.01	1.57	1.59	1.33

If we analyse only the results for the first generated solution (see Table 2), it can be inferred that the trade-off difference is smaller. For instance, the computation time for the first solution (the best one) is almost the same between C1 and C2, and the difference in the percentage of correct mappings is also small, only 5.13%. Probably the

best alternative is choosing configuration C3, which only loses 5.52% of accuracy and is more or less 300 milliseconds faster (these are average values for the 75 problems).

Table 2. Experimental results obtained for the first solution presented by REBUILDER, and for the three problem sets (average of values for P20, P50 and P80)

	C1	C2	C3	C4
Average of Computation Time (milliseconds)	4502	4433	4239	4287
Average of Percentage of Correct Mappings (%)	96.22	91.09	90.63	86.18
Average of Number of Mappings	5.01	5.01	5.01	4.90
Average Distance of Mapped Objects (is-a links)	0.47	0.81	0.90	1.47
Average Depth of MSCA (is-a links)	4.79	4.60	4.56	4.32
Average Number of Mappings x Correct Percentage	4.82	4.57	4.54	4.23

4 Discussion

This paper presents an approach for combining CBR retrieval with analogical retrieval using a general ontology for object categorization. One of the advantages of combining CBR and Analogy in a general CASE tool is that different types of designers can choose the reasoning most fit to their purposes. For instance, novice designers will certainly profit from CBR suggestions that help the exploration of semantic similar designs. On the other hand, experienced designers want more bold solutions, giving preference to optimising ones, using analogy.

From the method point of view, using case retrieval to get the analogue candidates is a way of optimising analogy retrieval. Experiments show that selection of analogue candidates based on a case retrieval algorithm with type and structural similarity is a good compromise solution between computing time and mapping quality.

Acknowledgements. This work was supported by POSI - Programa Operacional Sociedade de Informação of Portuguese Fundação para a Ciência e Tecnologia and European Union FEDER, under contract POSI/33399/SRI/2000, and by program PRAXIS XXI.

References

1. Kolodner, J., *Case-Based Reasoning*. 1993: Morgan Kaufman.
2. Gentner, D., *Structure Mapping: A Theoretical Framework for Analogy.* Cognitive Science, 1983. **7**(2): pp. 155–170.
3. Plate, T., *Distributed Representations and Nested Compositional Structure*, in *Graduate Department of Computer Science*. 1994, University of Toronto: Toronto.
4. Crean, B.P. and D. O'Donoghue. *Features of Structural Retrieval*. in *IASTED – International Symposia, Applied Informatics*. 2001. Innsbruck, Austria.
5. Smyth, B. and P. Cunningham. *Déjà Vu: A Hierarchical Case-Based Reasoning System for Software Design*. in *10th European Conference on Artificial Intelligence (ECAI'92)*. 1992. Vienna, Austria: John Wiley & Sons.

6. Maiden, N. and A. Sutcliffe, *Exploiting Reusable Specifications Through Analogy.* Communications of the ACM, 1992. **35**(4): pp. 55–64.
7. Rumbaugh, J., I. Jacobson, and G. Booch, *The Unified Modeling Language Reference Manual*. 1998, Reading, MA: Addison-Wesley.
8. Miller, G., *et al.*, *Introduction to WordNet: an on-line lexical database.* International Journal of Lexicography, 1990. **3**(4): pp. 235–244.

Combination Methods for Improving the Reliability of Machine Translation Based Cross-Language Information Retrieval

Gareth J.F. Jones and Adenike M. Lam-Adesina

Department of Computer Science, University of Exeter
Exeter, EX4 4PT, United Kingdom
{G.J.F.Jones,A.M.Lam-Adesina}@exeter.ac.uk

Abstract. Cross-Language Information Retrieval (CLIR) is an important topic in the increasingly multilingual environment of online information. Experiments using the standard CLEF 2001 bilingual task show that Machine Translation (MT) can provide effective search topic translation for CLIR, and that retrieval performance can be improved and made more reliable by applying a combination of pseudo-relevance feedback, corpus methods and data merging.

1 Introduction

The enormous growth in the availability of online documents in recent years has greatly increased the need for automated tools for information searching among unstructured document collections. One of the problems encountered in searching for information is that documents may be in different languages. Cross-Language Information Retrieval (CLIR) research seeks to provide part of the solution to this problem by searching for documents from a collection in one or more languages using search requests in a different language. One of the key issues for CLIR is translation. In this study we explore the use of easily available commercial Machine Translation (MT) systems for search topic translation [1].

This paper reports a study using MT systems for the European Cross-Language Evaluation Forum (CLEF) 2001 English language bilingual retrieval task [2]. We report results for French, German, Italian and Spanish topics, along with monolingual English topics for comparison. Two translation systems were used in the investigation: the online Babelfish (BF) translation system (available at `http://babelfish.altavista.com`) based on *SYSTRAN*, and *Globalink Power Translator Pro (PTP) Version: 6.4*. An important part of this study is an analysis of methods to combine the outputs of these MT systems. In addition, a number of methods have been shown to improve retrieval performance for CLIR. In this paper we investigate the use of Pseudo-Relevance Feedback (PRF) [1] and "pilot" searching using a large document collection. The results reported here show better overall average precision performance figures than the best results of official submissions to the CLEF 2001 workshop [2].

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 190–196, 2002.

The remainder of this paper is organised as follows: Section 2 reviews the information retrieval methods used, Sect. 3 describes the pilot search strategy, Sect. 4 describes the data combination methods, Sect. 5 gives the experimental results and analysis, and Sect. 6 concludes the paper.

2 Information Retrieval Methods

The experimental system used here is the City University research distribution of the Okapi system [3]. The documents and search topics were first processed to remove common stop words from a list of around 260 words, suffix stripped to encourage matching of different word forms, and further indexed using a small set of synonyms.

Term Weighting. Following preprocessing document terms are weighted using the Okapi BM25 *combined weight* [3], which combines collection frequency weighting ($cfw(i) = logN/n(i)$, where N is the total number of documents in the collection and $n(i)$ is the number of documents containing i), within document term frequency and document length compensation. (Two constants $K1$ and b are selected empirically for BM25.) A document matching score is computed by summing the weights of terms appearing in the query and the document. The documents are then returned in order of decreasing matching score.

Pseudo-relevance Feedback. Pseudo Relevance Feedback (PRF) assumes a number of top ranked documents from the initial search to be relevant. PRF selects a number of expansion terms which are then added to the topic prior to a further retrieval run. We adopt a PRF approach based on document summaries described in [4].

3 Pilot Searching

If additional documents are available these can be used in a "pilot searching" strategy to select expansion terms for the initial query. This larger data set enables more accurate estimates of collection parameters to be determined. The CLEF 2001 English collection of around 100,000 documents, taken from the LA Times, is a subset of the document collection used for the NIST Text REtrieval Conference (TREC) ad hoc tasks at TREC-8 [4] which consisted of around 500,000 documents. Our experiments used the TREC-8 ad hoc document set itself as a pilot collection, and the resulting expanded queries are then applied to the CLEF test collection.

The pilot searching procedure is carried out as follows: run the unexpanded query on pilot collection using BM25 without feedback; extract terms from the top R assumed relevant documents; rank the extracted terms [4]; select the desired number of expansion terms; add the terms to the topic; and perform a further retrieval run on test collection. N and $n(i)$ can either be taken from the pilot collection or from the test collection. Our investigation compares results

for BM25 with the $cfw(i)$ calculated using the test collection and the pilot collection. It further extends our earlier investigations of summary-based PRF by examining the effect of replacing $cfw(i)$ with the more sophisticated relevance weight $rw(i)$ [3], which incorporates a relevance component into $cfw(i)$.

4 Combination Methods

MT systems produce a single "best" translation which takes into account the available context. However, MT systems make translation errors, and even if the translation is accurate, the words appearing in the translated topic will be limited to those in the lexicon of the MT system. Combination of evidence from multiple information sources has been shown to be useful for text retrieval in TREC tasks [5]. In this study we explore the use of these ideas to combine the output of our two different MT systems using one of two techniques: *data fusion* and *query combination* [5].

Data Fusion. In data fusion ranked lists of retrieved documents from multiple retrieval systems are merged to form a single list. The ranked document lists produced independently by topics translated using BF and PTP were combined by adding the corresponding query-document matching scores from the two lists and forming a new re-ranked list using the composite scores.

Query Combination. In query combination multiple statements of a user information need are merged into a single query statement for application to a retrieval system. The translated queries produced by the two MT systems were combined into a single representation to score against the document archive.

Although ostensibly very similar the differences in these procedures will lead to different retrieval results. Our experimental investigation aims to establish if one method is clearly more appropriate for the task.

5 Experimental Results

This section describes results from our investigations for the CLEF 2001 bilingual task. The CLEF document collection consists of the test document set, 50 search topics available as natural manual translations in a variety of languages, and manual relevance assessments formed using standard pooling procedures [2]. The experiments here all use the Title and Description field of the CLEF topics. Results are shown in each case for precision at a cutoff of 10 documents retrieved (P10) and average precision (AvP).

Parameters for the Okapi system and our document summariser were selected using the CLEF 2000 bilingual test collection. Okapi parameters were set as: $K1 = 1.0$ and $b = 0.5$. For PRF, 5 documents assumed relevant, document summaries used the best 4 sentences. Potential expansion terms were ranked by assuming the top 20 documents were relevant. 20 expansion terms added to the original topic. Original topic terms upweighted by a factor of 3.5 relative to expansion terms.

Single Translation Results. Tables 1 and 2 show results for the two MT systems. The best average precision achieved for each language pair is underlined. It can be seen that both the MT systems give reasonable CLIR performance relative to the monolingual English baseline, although there is considerable variablity. While French topic translation using BF is by far the best overall, retrieval after topic translation using PTP generally shows more consistent behaviour. In all cases feedback improves the absolute average precision over the baseline figures.

For pilot collection selection of expansion terms, 5 documents were assumed relevant, number of documents for expansion term ranking 20, and the summary length now 6 sentences, as used in [4]. The term weights used in the test collection run continue to be derived from the test collection itself. The results for BF show that selection using the pilot collection gives a small improvement in performance for 2 languages and a small decrease for the other 2. However, for PTP term selection from the pilot collection is shown to be beneficial in all cases. BF results for pilot searching with the $cfw(i)$ now calculated from the pi-

Table 1. Summary of retrieval results for BF topic translation

		Topic Language				
		English	French	German	Italian	Spanish
Bl. Test Coll.*cfw(i)*	P10	0.406	0.366	0.330	0.298	0.353
Bl. Test Coll.*cfw(i)*	AvP	0.484	0.473	0.398	0.375	0.389
Test Coll. Select +	P10	0.400	0.366	0.336	0.320	0.394
Test Coll. *cfw(i)* Fb.	AvP	0.517	0.489	0.415	0.395	0.423
Pilot Select +	P10	0.434	0.381	0.330	0.296	0.357
Test Coll. *cfw(i)* Fb.	AvP	0.524	0.502	0.419	0.391	0.418
Pilot Select +	P10	0.447	0.400	0.323	0.310	0.372
Pilot *cfw(i)* Fb.	AvP	0.568	0.512	0.430	0.422	0.436
Pilot Select +	P10	0.460	0.434	0.345	0.321	0.387
Pilot *rw(i)* Fb.	AvP	0.582	0.526	0.420	0.426	0.448

Table 2. Summary of retrieval results for PTP topic translation

		Topic Language				
		English	French	German	Italian	Spanish
Bl. Test Coll.*cfw(i)*	P10	0.406	0.368	0.349	0.364	0.383
Bl. Test Coll.*cfw(i)*	AvP	0.484	0.438	0.439	0.427	0.417
Test Coll. Select +	P10	0.400	0.402	0.381	0.396	0.411
Test Coll. *cfw(i)* Fb.	AvP	0.517	0.466	0.456	0.432	0.419
Pilot Select +	P10	0.434	0.387	0.381	0.377	0.400
Test Coll. *cfw(i)* Fb.	AvP	0.524	0.476	0.499	0.445	0.453
Pilot Select +	P10	0.447	0.387	0.387	0.366	0.385
Pilot *cfw(i)* Fb.	AvP	0.568	0.455	0.489	0.437	0.451
Pilot Select +	P10	0.460	0.412	0.404	0.385	0.389
Pilot *rw(i)* Fb.	AvP	0.582	0.496	0.512	0.464	0.463

lot collection show that this method is generally effective for all language pairs. The results for PTP are rather more surprising; in this case replacing the $cfw(i)$ values produces a reduction in CLIR performance for all language pairs. This is a rather unexpected result and is investigated further later in the paper. For BF translation use of the $rw(i)$ generally produces a further small improvement in average precision. The result for PTP is much clearer, the reduction in performance from introducing pilot collection $cfw(i)$ between is now reversed. In terms of MT quality, SYSTRAN (BF) French-English is one of the most longstanding and best developed MT systems, and SYSTRAN Italian-English generally found to be one of the less reliable translation pairs [private communications]. It is interesting to see these general observations in translation quality reflected in the retrieved results for BF topic translation.

Data Combination Experiments. Tables 3 and 4 show retrieval performance for data fusion and query combination respectively. Comparing baseline data fusion results with single MT baseline, it can be seen that data fusion method generally offers improvement. The result for German is particularly good, giving the same value as the English baseline. Baseline query combination results are generally less good than those for data fusion. Results for data fusion with PRF using only the test collection show improvement in average precision over the data fusion baseline, except again for German. In this case query combination results are generally better than data fusion. Data fusion results for pilot searching term selection with $cfw(i)$ weights taken from the test collection are generally better than test collection expansion selection. The corresponding query combination results are again generally similar to those for data fusion with neither method obviously being generally superior. For term selection and $cfw(i)$ using the pi-

Table 3. Summary of retrieval results for simple score addition data fusion

		Topic Language				
		English	French	German	Italian	Spanish
Test Coll.BF Bl	P10	0.406	0.366	0.330	0.298	0.353
	AvP	0.484	0.473	0.398	0.375	0.389
Test Coll.PTP Bl	P10	0.406	0.368	0.349	0.364	0.383
	AvP	0.484	0.438	0.439	0.427	0.417
Bl.	P10	0.406	0.385	0.394	0.387	0.385
Test Coll. *cfw(i)*	AvP	0.484	0.479	0.484	0.426	0.423
Test Coll. Select +	P10	0.400	0.415	0.396	0.400	0.413
Test Coll. *cfw(i)* Fb.	AvP	0.517	0.489	0.476	0.451	0.426
Pilot Coll. Select +	P10	0.434	0.398	0.413	0.381	0.400
Test Coll. *cfw(i)* Fb.	AvP	0.524	0.502	0.524	0.446	0.462
Pilot Coll. Select +	P10	0.447	0.409	0.413	0.381	0.398
Pilot Coll. *cfw(i)* Fb.	AvP	0.568	0.497	0.515	0.465	0.460
Pilot Coll. Select +	P10	0.460	0.445	0.432	0.379	0.400
Pilot Coll. *rw(i)* Fb.	AvP	0.582	0.533	0.518	0.468	0.465

Table 4. Summary of retrieval results for query combination

		Topic Language				
		English	French	German	Italian	Spanish
Test Coll.BF Bl	P10	0.406	0.366	0.330	0.298	0.353
	AvP	0.484	0.473	0.398	0.375	0.389
Test Coll.PTP Bl	P10	0.406	0.368	0.349	0.364	0.383
	AvP	0.484	0.438	0.439	0.427	0.417
Bl.	P10	0.406	0.372	0.375	0.372	0.375
Test Coll. *cfw(i)*	AvP	0.484	0.455	0.457	0.440	0.403
Test Coll. Select +	P10	0.400	0.404	0.394	0.385	0.409
Test Coll. *cfw(i)* Fb.	AvP	0.517	0.498	0.406	0.469	0.421
Pilot Coll. Select +	P10	0.434	0.396	0.398	0.357	0.394
Test Coll. *cfw(i)* Fb.	AvP	0.524	0.509	0.488	0.457	0.442
Pilot Coll. Select +	P10	0.447	0.381	0.364	0.317	0.366
Pilot Coll. *cfw(i)* Fb.	AvP	0.568	0.452	0.464	0.410	0.428
Pilot Coll. Select +	P10	0.460	0.402	0.347	0.285	0.389
Pilot Coll. *rw(i)* Fb.	AvP	0.582	0.489	0.440	0.400	0.442

Table 5. Summary of average precision retrieval results for query combination with common translation terms double weighted

		Topic Language				
		English	French	German	Italian	Spanish
Pilot Coll. Select +	P10	0.447	0.430	0.381	0.313	0.398
Pilot Coll. $cfw(i)$ Fb.	AvP	0.568	0.495	0.479	0.404	0.462

lot collection, the French result is lower than the excellent result achieved for BF translation on its own shown in Table 1 (data fusion with the much poorer PTP French result is probably responsible for this). Comparing the results to those taking $cfw(i)$ from the test collection, performance is similar or worse. The results for query combination are more surprising, being significantly worse than those for test collection $cfw(i)$. The results for data fusion using $rw(i)$ are the best achieved so far, showing both better overall performance and consistency than the individual translation results. Query combination results are again much poorer than those for data fusion. Overall the best query combination results are found using test collection $cfw(i)$.

Analysis of the retrieval effect of data fusion vs query combination shows that the main difference between them arises from the double weighting in data fusion of terms translated the same by both MT systems. The observed results suggest that giving these terms additional weight may be beneficial. In order to test this theory we repeated the query combination pilot collection $cfw(i)$ run doubling the weights of the terms common to both translations. The results for this new experiment are shown in Table 5. The results show that upweighting

the common terms does indeed tend to improve query combination results, only the Italian result is not improved by this technique.

6 Concluding Remarks and Further Work

This paper has presented our results for the CLEF 2001 bilingual retrieval task. The results indicate that good retrieval results can be achieved using commercial MT systems. The application of PRF, pilot collection and, the data fusion and query combination methods with alternative topic translations is generally shown to have a positive impact on results. When combining all the evidence together it was found that use of data fusion with pilot searching and relevance weighting gave the best overall retrieval results, generally with greater consistency than taking either MT system on its own.

References

[1] G. J. F. Jones, T. Sakai, N. H. Collier, A. Kumano, and K. Sumita. A Comparison of Query Translation Methods for English-Japanese Cross-Language Information Retrieval. In *Proceedings of ACM SIGIR*, pages 269–270, San Francisco, 1999.

[2] Carol Peters, editor. *Workshop of the Cross-Language Evaluation Forum, CLEF 2001*, Darmstadt, September 2001. Springer.

[3] S. E. Robertson, et al. Okapi at TREC-3. In D. K. Harman, editor, *Proceedings of the TREC-3)*, pages 109–126. NIST, 1995.

[4] A. M. Lam-Adesina and G. J. F. Jones. Applying Summarization Techniques for Term Selection in Relevance Feedback. In *Proceedings of ACM SIGIR*, pages 1–9, New Orleans, 2001.

[5] N. J. Belkin, et al. Combining the Evidence of Multiple Query Representations for Information Retrieval. *Info. Processing and Management.*, 31:431–448, 1995.

A System for Music Information Retrieval

Orla Lahart and Colm O'Riordan

Department of IT, NUI, Galway
orlalahart@hotmail.com, colmor@geminga.nuigalway.ie

Abstract. Information Retrieval, and in part Music Information Retrieval, has come to the fore in recent years as a research area. In this paper we outline the design of a system which allows for the retrieval of tunes from a music database. We describe algorithms and implementation of techniques to allow for the retrieval of tunes based on the occurrence of phrases or sub-phrases together with algorithms for the comparison of tunes via the vector space model.

1 Introduction

Much research has been carried out in the area of Information Retrieval (IR) and the specialisation of IR, Multimedia Information Retrieval (MMIR), in recent years. Two typical features of multimedia data is that it is large in size and the data is often unstructured. Music Information Retrieval (MIR) is a specialised subfield of MMIR wherein one attempts to provide techniques to allow for the accurate retrieval of tunes given a user's query.

In this paper we discuss the design and implementation of a Music Retrieval system which supports querying for tunes based on phrases and prefixes. The system also supports retrieval and clustering of tunes using a vector space model. The first section of this paper outlines the area of IR. This is followed by an introduction to the main concepts of MMIR, and more specifically MIR. The overall design and architecture of the system is then outlined. The paper concludes with results to date and possible future work.

2 Information Retrieval

Information Retrieval (IR) deals with the representation, storage, organisation and retrieval of information items [2]. The area of IR has expanded over the past number of years, primarily this is due to the abundance of information available, particularly on the World Wide Web. Nowadays, research in IR includes document classification and categorisation, modeling, filtering and indeed the specialisation, Multimedia Information Retrieval (MMIR). Many IR models exist. The three most commonly used models, which are sometimes referred to as the classical models of IR, are the Boolean model [2], the probabilistic model [10] and the vector model (where each document is represented as a vector and the similarity between two documents is evaluated by calculating the cosine of the angle between them) [11].

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 197–202, 2002.

2.1 Multimedia Information Retrieval

The main goals of a Multimedia Information Retrieval (MMIR) system, as with any IR system, is to retrieve information efficiently. The system must have some capability for dealing with approximate or fuzzy queries. Once the query is formulated by the user the system must parse and compile the query into an internal format. The relevance of a stored object is quantified by its distance from the query object.

A major problem related to multimedia data is how to represent it within the system. An object should be described by a set of its features. These features can be automatically selected, manually selected by the user, or indeed a combination of both. Feature extraction can not be precise, therefore a weight is often associated with a feature which specifies the degree of certainty associated with it. Once the features have been extracted multi-attribute or spatial access methods (such as *R-trees* and *R*-trees*) can be used to search for them. By representing an object by f features each object can be mapped to a point in f-dimensional space. This mapping provides us with a way of clustering objects.

3 Previous Systems

MIR systems have been developed over the past number of years. There are a wide variety of systems available. However, most require that the user is in some way musical. Many systems employ *midi* as the internal representation of music. However, there are some exceptions [13]. In addition, some systems store the note direction of a melody as opposed to the *midi* value [5,7]. Traditional IR techniques have been adopted by some systems [3,8]. With music it can be difficult to calculate the similarity between tunes, many systems employ the algorithm proposed by [1]. Such systems include [5], [7] and [9].

In traditional IR systems, choosing the alphabet by which to represent documents is very important. This is also true for MIR systems. Previous research suggests that it is often the case that a smaller alphabet can provide more effective results. It is also believed that it is the shape, or contour, of a tune that is usually recognised or remembered. Therefore, a lot of work has been done on reducing the size of the alphabet. This idea was first introduced by Parson, each note of a piece of music can only be higher, lower or the same as the previous note. Parson's concept can be represented in a number of different ways. Lap and Kao use an alphabet consisting of $\rightarrow$, $\nearrow$ and $\searrow$ [6]. Therefore, the following contour represents the beginning of Happy Birthday. An asterix indicates the first note.

Happy Birthday to you Happy Birthday to you
$* \rightarrow \nearrow \searrow \nearrow \searrow \searrow \rightarrow \nearrow \searrow \nearrow \searrow$

If the concepts in Parson's code are exploited further the same approach can be taken when representing the rhythm of a tune. If the note has the same duration as the previous note it will be represented as $\frac{1}{1}$, if the note is twice as long as the

previous note it can be expressed as $\frac{2}{1}$. There is one drawback with using this method: we loose other characteristics of rhythm such as accents. An advantage to note duration is that just as melodic contour is invariant to transposition (frequency scaling), note duration ratio sequence is invariant to time-scaling. This accommodates imperfect memories.

4 Music Information Retrieval System Architecture

The system described in this paper is a fully functional MIR system. It encompasses techniques adopted from the field of IR. A user can interact with the system in various ways. The system provides functionality for fixed length queries, prefix queries and approximate queries. The system also allows for the clustering of tunes. Each tune is represented as a set of features; these features are extracted from *midi* files.

4.1 Indexing

In MIR the selection of index terms is quite complex. In this system, each contour is divided into overlapping sub-contours of a fixed length. This is to aid querying. Parson's code is used for indexing. Due to the structure of the index terms, it is important to select a data structure that would provide an effective storage mechanism and more importantly provide a means for efficient searching. With this intention it was decided that the index terms would be stored using a trie. Tries are extremely efficient data structures where construction is *O(n)* worst case and searching is *O(m)*, where m is the length of the search string [4].

4.2 Vector Space Model

The vector space model provides the backbone for this system. To fully exploit the effectiveness of the model the traditional structure was adapted to suit the needs of this system. It was decided that each tune would be represented by a set of features. It is important to strike a balance: too many features can result in the process being unduly complicated and computationally expensive while too few features can make it difficult to distinguish between objects. A total of six features were selected; common melodic pattern, common rhythmic pattern, key signatures, time signatures, average pitch and surprise interval.

Features. Common melodic pattern was chosen as a feature because most often it is the melody of a tune that a listener will first notice and remember. This is because it contains frequently re-occurring melodic motifs. Firstly, the melody is extracted from the *midi* file. Uidenbogerd and Zobel outline several algorithms for extracting the melody [12]. The algorithm, which performed best over all tests, was that which chooses the top note of all notes starting at any instance (the *Skyline* algorithm). Similar motivations exist for the selection of common

rhythmic patterns. This pattern is also represented using Parson's code. Both of these features are stored using a trie.

At a basic level, music can be categorised as that which empowers us or instills emotions of melancholy. This is dependent on the mood of the music. Often tunes that are written in a minor key and have a slow tempo are connected with emotions of sadness. This suggests that a listener can distinguish between tunes based on time signature and key signature.

The range of notes used within a tune is another distinguishing factor, thus average pitch has been chosen as a feature. The last feature selected was the surprise interval feature. Research suggests that the most commonly occurring intervals include single small intervals (-3 to +3 semitones) or sequences or repeats of small intervals are common. As a result, intervals with large magnitudes are less frequently used and thus create a sense of "surprise" [6]. This feature can be represented as a Boolean value.

Each tune is represented as a set of these features. The similarity between two tunes is calculated based on the similarity of the features of each tune. This is achieved through the use of the vector model. Where each vector comprises the features associated with that tune. Vector-based features (i.e. common melodic pattern and common rhythmic pattern) must be treated independently. The user can choose to view results as a ranked list or as a cluster.

4.3 Neighbourhood Clustering Algorithm

In addition to clustering tunes in relation to a selected tune the system must also cluster tunes in relation to each other. Each tune is represented as a cluster. The two closest clusters are selected first. These are merged to form a new cluster. The centroid of the vectors representing the tunes is used to represent the cluster. This process continues until all clusters have been merged. A tree structure representing the clusters will result. Therefore, all tunes that are similar will be mapped close together in space. The system uses the similarity correlation as the distance function. With this approach the user can identify inter-relationships between tunes.

5 Results

Experiments have been conducted using two (small) databases of *midi* files which were down-loaded form the World Wide Web. One database consists of popular music, the other is a collection of classical music. The testing of exact and prefix queries is quite straightforward. It is a form of data retrieval and the system will retrieve correctly or not. Approximate queries are slightly more complex to test. One must quantify the accuracy of returned results. The approach adopted (i.e. map tunes to feature space) is noisy in so far as some data may be lost, and, furthermore, queries may not be of an exact nature.

Results obtained from both fixed length and prefix queries proved successful. Research to date suggests that twelve notes are required to identify most

tunes [7]. However, testing of this system showed that six notes were sufficient. Prefix queries were slightly more successful than fixed length queries. It must be remembered that the version of the tune in the database may be different to that known by the user. In addition, due to the use of the *Skyline* algorithm there may be some inaccuracies in the melody extracted. Prefix queries can deal better with this.

Evaluating the validity of approximate queries is more complex. This is due to the lack of a recognised standard test collection for MIR similar to those which exist in traditional IR, for example TREC. Due to this, there was a need for the development of a test collection. This collection consists of the tunes from the database as well as tunes which contain slight modifications. A number of experiments were carried out, these experiments proved the validity of the techniques and algorithms adopted. Tunes when compared to their copies (which contained subtle modifications) proved to be highly correlated, as expected. In addition, tunes which would not sound similar to the user had a very low correlation. In general, popular tunes had a higher correlation with tunes of the same genre than with classical tunes. It was also noted that some features had a more discriminatory effect than others. This provides strong evidence to indicate the validity and accuracy of the algorithms employed in this system. It is believed that both common melodic pattern and common rhythmic pattern had the most discriminatory effects. It was also noted that by increasing the length of the aforementioned features from four to six proved very effective. On experimentation, it was identified that increasing the weight associated with particular features improved the overall performance of the system.

6 Conclusion

The system outlined in this paper is a fully functional, extendible, robust MIR system. The system provides functionality for fixed length, prefix and approximate queries. To the author's best knowledge this degree of functionality is not available in other systems. The features selected to represent each tune are unique to this system. A number of IR techniques have been adopted, namely indexing techniques, querying techniques and clustering techniques. Being musically inclined is not a prerequisite for interaction with this system, as a users information need is represented using Parson's code. Results to date highlight the validity of the approaches and techniques adopted. Future work could include the employment of additional features and the modification of existing features and further empirical analysis following the development of a more comprehensive test collection.

References

1. Ricardo A. Baeza-Yates and C. H. Perleberg. Fast and practical approximate string matching. In *Combinatorial Pattern Matching, Third Annual Symposium*, pages 185–192, 1992.

2. Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Publishing Company, Reading MA, 1999.
3. Stephen J. Downie and Michael Nelson. Evaluation of a simple and effective music information retrieval method. In *Proceedings of ACM-SIGIR Conference*, 2000.
4. William B. Frakes and Ricardo Baeza-Yates. *Information Retrieval: data structures and algorithms*. Prentice Hall, Englewood Cliffs, N.J., 1992.
5. Asif Ghias, Jonathan Logan, David Chamberlin, and Brian C. Smith. Query by humming, musical information retrieval in an audio database. In *ACM Multimedia*, 1995.
6. Yip Chi Lap and Ben Kao. A study on musical features for melody databases. In *Databases and Expert Systems Applications*, 1999.
7. Rodger J. McNab, Llyod A. Smith, D. Bainbridge, and Ian H. Witten. The new zealand digital library melody index, 1997.
8. Massimo Melucci and Nicola Orio. Smile: a system for content-based musical information retrieval environment. `citeseer.nj.nec.com/42002.html`.
9. Omras. Omras – online music recognition and searching. `http://www.omras.org`, 2000.
10. S. E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *The Journal of the American Society for Information Sciences*, 27(3):129–146, 1976.
11. Gerard Salton and M.E. Leek. Computer evaluation of indexing and text processing. *Journal of the ACM*, 15(1):8–36, 1968.
12. Alexandra L. Uitdenbogerd and Justin Zobel. Manipulation of music for melody matching. In *ACM Multimedia*, 1998.
13. Matt Welsh, Nikita Borisov, Jason Hill, Rob von Behren, and Alec Woo. Querying large collections of music for similarity. `http://www.citeseer.nj.nec.com/weldh99querying.html`, 1999.

A New Bayesian Network Structure for Classification Tasks

Michael G. Madden

Department of Information Technology
National University of Ireland, Galway, Ireland
michael.madden@nuigalway.ie

Abstract. This paper introduces a new Bayesian network structure, named a *Partial Bayesian Network* (PBN), and describes an algorithm for constructing it. The PBN is designed to be used for classification tasks, and accordingly the algorithm constructs an approximate Markov blanket around a classification node. Initial experiments have compared the performance of the PBN algorithm with Naïve Bayes, Tree-Augmented Naïve Bayes and a general Bayesian network algorithm (K2). The results indicate that PBN performs better than other Bayesian network classification structures on some problem domains.

1 Introduction

Bayesian networks graphically represent the joint probability distribution of a set of random variables. A Bayesian network structure (B_S) is a directed acyclic graph where the nodes correspond to domain variables x_1, …, x_n and the arcs between nodes represent direct dependencies between the variables. Likewise, the absence of an arc between two nodes x_1 and x_2 represents that x_2 is independent of x_1 given its parents in B_S. Following the notation of Cooper and Herskovits [5], the set of parents of a node x_i in B_S is denoted as π_i. The structure is annotated with a set of conditional probabilities (B_P), containing a term $P(x_i=X_i|\pi_i=\Pi_i)$ for each possible value X_i of x_i and each possible instantiation Π_i of π_i.

The objective of the research presented in this paper is to develop a methodology for induction of a Bayesian network structure that is specifically geared towards classification tasks. This structure, which is named a *Partial Bayesian Network* (PBN), should include only those nodes and arcs that affect the classification node.

2 Induction of Bayesian Networks

In the work described here, the framework developed by Cooper and Herskovits [5] for induction of Bayesian networks from data is used. This is based on four assumptions: (1) All variables are discrete and observed; (2) Cases occur independently, given a belief network model; (3) No cases have variables with missing values; (4) Prior probabilities of all valid network structures are equal.

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 203–208, 2002.

Let Z be a set of n discrete variables, where a variable x_i in Z has r_i possible values: $(v_{i1}, \ldots, v_{ir_i})$. Let D be a database of m cases, each with a value for every variable in Z. Let w_{ij} denote the jth unique instantiation of π_i relative to D, where there are q_i such instantiations. Let N_{ijk} be defined as the number of cases in D in which variable x_i has the value v_{ik} and π_i is instantiated as w_{ij}. Let N_{ij} be the sum of N_{ijk} over all instantiations of x_i. Cooper and Herskovits [5] derive the following equation:

$$g(i,\pi_i) = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \tag{1}$$

This is the basis of their K2 algorithm, which takes as its input an ordered list of n nodes and a database D containing m cases. Its output is a list of parents for each node. In a single iteration of K2, an arc is added to node i from the node z that maximizes $g(i,\pi_i \cup \{z\})$. If $g(i,\pi_i) > g(i,\pi_i \cup \{z\})$, no arc is added [5].

To calculate conditional probabilities, let θ_{ijk} denote the conditional probability that a variable x_i in B_S has the value v_{ik}, for some k from 1 to r_i, given that π_i is instantiated as w_{ij}. Then, given the database D, the structure B_S and the assumptions listed earlier (denoted ξ), the expected value of θ_{ijk} is given by [5]:

$$E[\,\theta_{ijk} | D; B_S; \xi\,] \;=\; \frac{N_{ijk} + 1}{N_{ij} + r_i} \tag{2}$$

3 Using a Bayesian Network for Classification

A noteworthy feature of Bayesian classifiers is their ability to accommodate noisy data: conflicting training examples decrease the likelihood of a hypothesis rather than eliminating it completely.

A Bayesian network may be used for classification as follows. Firstly, assume that the classification node x_c is unknown and all other nodes are known. Then, for every possible instantiation of x_c, calculate the joint probability of that instantiation of all nodes given the database D, as follows [5]:

$$P(x_1 = X_1, \ldots, x_n = X_n) = \prod_{i=1}^{n} P(x_i = X_i | \pi_i = \Pi_i) \tag{3}$$

By normalizing the resulting set of joint probabilities of all possible instantiations of x_c, an estimate of the relative probability of each is found.

4 Related Research

The simplest form of Bayesian classifier, known as *Naïve Bayes*, was shown by Langley *et al.* [9] to be competitive with Quinlan's popular C4.5 decision tree classifier [12]. Naïve Bayes is so called because it makes the two following, often unrealistic,

assumptions: (1) All other variables are conditionally independent of each other given the classification variable; (2) All other variables are directly dependent on the classification variable. Represented as a Bayesian network, a Naïve Bayes classifier has a simple structure whereby there is an arc from the classification node to each other node, and there are no arcs between other nodes [6].

Researchers have examined ways of achieving better performance than Naïve Bayes by relaxing these assumptions. Friedman *et al.* [6] analyze *Tree Augmented Naïve Bayes* (TAN), which allows arcs between the children of the classification node x_c, thereby relaxing the first assumption above. In their approach, each node has x_c and at most one other node as a parent, so that the nodes excluding x_c form a tree structure. They use a *minimum description length* metric rather than the Bayesian metric used in this paper (though they note Heckerman's observation [7] that these are asymptotically equivalent). To find arcs between the nodes, they use an algorithm first proposed by Chow and Liu [4] for learning tree-structured Bayesian networks.

Langley and Sage [10] consider an alternative approach called *Selective Naïve Bayes* (SNB), in which a subset of attributes is used to construct a Naïve Bayes classifier. By doing this, they relax the second of the two assumptions listed above. Kohavi and John [8] improve on this by using a wrapper approach to searching for a subset of features over which the performance of Naïve Bayes is optimized.

Cheng and Greiner [3] evaluate the performance of two other network structures. The first is *Bayesian Network Augmented Naïve Bayes* (BAN), in which all other nodes are direct children of the classification node, but a complete Bayesian network is constructed between the child nodes. The second is the *General Bayesian Network* (GBN), in which a full-fledged Bayesian network is used for classification. After constructing the network, they delete all nodes outside the Markov blanket prior to using the network for classification. They use an efficient network construction technique based on conditional independence tests [2]. They report good results with the BAN and GBN algorithms compared with Naïve Bayes and TAN, particularly when a wrapper is used to fine-tune a threshold parameter setting.

5 Partial Bayesian Networks for Classification

The motivation behind this research is similar to that of the authors already discussed: to construct Bayesian network structures that are specifically geared towards classification tasks. The method presented here seeks to directly construct an approximate *Partial Bayesian Network* (PBN) for the Markov blanket around the classification node. As described by Pearl [11], the Markov blanket of a node x is the union of x's direct parents, x's direct children and all direct parents of x's direct children. The Markov blanket of x is one of its Markov boundaries, meaning that x is unaffected by nodes outside the Markov blanket.

The procedure for construction of a PBN involves three steps. In the first step, every node $x_i \in Z-\{x_c\}$ is tested relative to x_c to determine whether it should be considered to be a parent or a child of x_c. If x_i is added as a parent of x_c, the overall probability of the network will change by a factor δ_p that is calculated as:

$$\delta_p = \frac{g(c;\pi_c \cup \{i\})}{g(c;\pi_c)} \tag{4}$$

Alternatively, if x_i is added as a child of x_c, the probability will change by δ_c:

$$\delta_c = \frac{g(i;\pi_i \cup \{c\})}{g(i;\pi_i)} \tag{5}$$

Accordingly, by testing whether $\delta_p > \delta_c$, x_i is added to either the set of x_c's parent nodes Z_P or its child nodes Z_C. However, if $max(\delta_p, \delta_c) < 1$, no arc is added; x_i is added to the set of nodes Z_N that are not directly connected to x_c.

At the end of the first step, having performed this calculation for each node in turn, a set of direct parents of x_c (Z_P), direct children (Z_C), and nodes not directly connected (Z_N) have been identified. It is noted that this procedure may be sensitive to the node ordering, since π_c changes as parent nodes are added. In ongoing work, this author is examining variations on the procedure that involve increased computational effort, to see whether they would improve accuracy significantly. For example, it is possible to iterate repeatedly over all nodes, each time adding the node with maximum δ_p.

The second and third steps are concerned with completing the Markov blanket by finding the parents of x_c's children. In the second step, parents are added to the nodes $x_i \in Z_C$ from a set of candidates $Z_P \cup Z_N$ using the K2 algorithm. In experiments to date, this has required less computation than a full invocation of K2, since the nodes have been partitioned into mutually exclusive sets of children and candidate parents. This partitioning also means that PBN does not require K2's node ordering.

In the third step, dependencies between the nodes in Z_C are found. Since children of x_c may be parents of other children of x_c, such dependencies fall within the Markov blanket of x_c. This step is performed by constructing a tree of arcs between the nodes in Z_C. This is similar to what is done in the TAN algorithm, except that it handles nodes having different sets of parents. Naturally, this is an approximation, as it can discover at most one additional parent for each node within the group.

6 Initial Experimental Results

Two datasets from the UCI machine learning repository [1] have been used for preliminary experiments: the relatively simple Wisconsin Breast Cancer dataset (683 examples, 10 discrete attributes, 2 classes) and the more complex Chess dataset for the King & Rook versus King & Pawn endgame (3196 examples, 36 binary/ternary attributes, 2 classes). For these datasets, the accuracy of the PBN algorithm was compared with that of Naïve Bayes, TAN and GBN algorithms, all of which were implemented using Cooper and Herskovits's inductive learning framework as described in Section 2 — the GBN algorithm was actually K2. In running K2, the node ordering of the original datasets was used, except that the classification node was placed first so that it could be included as a parent of any other node. Learning curves were constructed, to compare the algorithms over a range of training set sizes. Figure 1 shows the results of these experiments. Each point on the graphs is an average over 10 runs.

For clarity, the variance of results is not shown; in all cases, the standard deviations were less than 1% for training set sizes above 20%, except for Naïve Bayes on the Chess dataset, where they were in the range 1%-1.5%.

For the Breast Cancer dataset, PBN performed about the same as K2 and Naïve Bayes. In Fig. 1, PBN appears to under-perform Naïve Bayes, but the difference is not significant; using a paired t-test to compare results on 20 runs with 2/3 of the data randomly selected for training, the difference was not significant at the 95% confidence level. Very similar results to ours are reported by Friedman *et al* [6] for Naïve Bayes on this dataset; they found that it outperformed TAN, SNB and C4.5 on this dataset. In fact, in our analysis, networks constructed by PBN and K2 were essentially Naïve Bayes structures with a small number of arcs added or removed.

The results are more decisive for the Chess dataset, also shown in Fig. 1. Here, the Naïve Bayes classifier did not perform well; one reason for this is that there appear to be significant correlations between attributes, as indicated by the way that TAN performs substantially better than Naïve Bayes. As the graph shows, K2 outperformed TAN and our PBN algorithm outperformed K2. The difference in results between PBN and K2 has been verified to be statistically significant at the 95% and 99% confidence levels, using a paired t-test. One theory to account for this is that PBN performed better than K2 because it is not constrained by node ordering, and because it may be able to find more subtle dependencies between variables; this will tested in the future by running K2 with a node ordering derived from PBN.

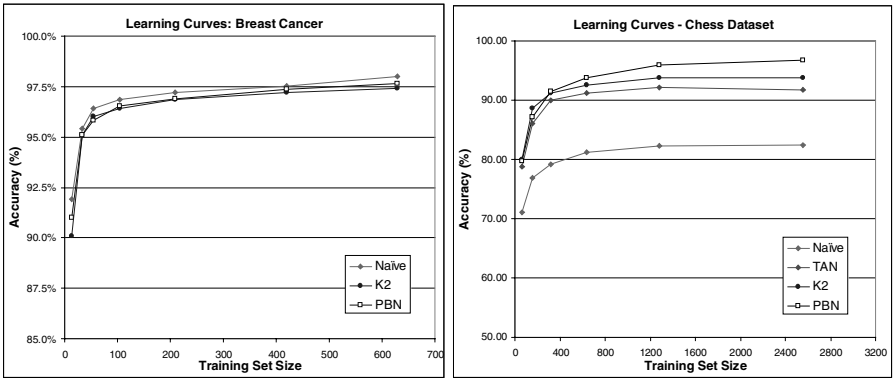


Fig. 1. Comparison of PBN with other Bayesian Classifiers

7 Conclusions & Future Work

This paper has presented a new Bayesian network structure for classification, called a *Partial Bayesian Network*, and a method for constructing the PBN, in which a Markov blanket is constructed around the classification node. Key features are:

- In the first step of constructing the PBN, all nodes are classified as either parents of the classification node, children of it, or unconnected to it. This contrasts with Naïve Bayes, TAN and BAN structures, where all nodes must be children of the classification node. It also contrasts with SNB, where a wrapper approach is taken to find which nodes are connected to the classification node.

- In the second and third steps of constructing the PBN, the only arcs added are to children of the classification node, so that an approximate Markov blanket around the classification node is constructed. This contrasts with GBN structures, in which arcs may be added outside of the Markov blanket but are not considered when using the GBN for classification.
- Unlike K2, the PBN algorithm does not require an ordering on the nodes.

Initial experimental results appear promising, in that PBN outperforms Naïve Bayes, TAN and GBN (where all are implemented using the framework of Cooper and Herskovits) on the moderately complex Chess dataset, and matches their performance in the simpler Wisconsin Breast Cancer dataset. Further experiments are required to evaluate the PBN method fully. It is also hoped to research whether the PBN approach could be improved by using a different scoring metric such as MDL or conditional independence testing. Finally, it is also hoped to investigate dynamic discretization of continuous variables while constructing the network. The structure of the PBN should facilitate this, as all variables are associated with the (necessarily discrete) classification node.

Acknowledgement. This work has been supported by NUI Galway's Millennium Research Programme.

References

1. Blake, C.L. & Merz, C.J., 1998. UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. University of California at Irvine.
2. Cheng, J., Bell, D.A. & Liu, W., 1997: Learning Belief Networks from Data: An Information Theory Based Approach. Proc. ACM CIKM '97
3. Cheng, J. & Greiner, R., 2001: Learning Bayesian Belief Network Classifiers: Algorithms and System. Proc. 14th Canadian Conference on Artificial Intelligence.
4. Chow, C.K. & Liu, C.N., 1968: Approximating Discrete Probability Distributions with Dependence Trees. IEEE Transactions on Information Theory, Vol. 14, 462–267.
5. Cooper, G.F. & Herskovits, E., 1992: A Bayesian Method for the Induction of Probabilistic Networks from Data. Machine Learning, Vol. 9, 309–347. Kluwer, Boston.
6. Friedman, N., Geiger, D. & Goldszmidt, M., 1997: Bayesian Network Classifiers. Machine Learning, Vol. 29, 131–163. Kluwer, Boston.
7. Heckerman, D, 1996: A Tutorial on Learning with Bayesian Networks. Technical Report MSR-TR-95-06, Microsoft Corporation, Redmond.
8. Kohavi, R. & John. G., 1997: Wrappers for Feature Subset Selection. Artificial Intelligence Journal, Vol. 97, No. 1-3, 273–324.
9. Langley, P., Iba, W. & Thompson, K., 1992: An Analysis of Bayesian Classifiers. Proc. AAAI-92, 223–228.
10. Langley, P. & Sage. S., 1994: Induction of Selective Bayesian Classifiers. Proc. UAI-94.
11. Pearl, J., 1988: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Francisco.
12. Quinlan, J.R., 1993: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco.

Evaluating Preference-Based Feedback in Recommender Systems*

Lorraine Mc Ginty and Barry Smyth

Smart Media Institute, University College Dublin, Dublin 4, Ireland
{lorraine.mcginty,barry.smyth}@ucd.ie

Abstract. Anecdotal evidence suggests that while real-world customer-buying models *can* be translated for online e-commerce scenarios, perhaps they should not be (because of the effort these models impose on the user). Currently the vast majority of recommender systems require the user to engage in a lengthy dialogue of information elicitation in order to locate the best product/service for them. We suggest that such lengthy dialogues are inappropriate in many e-commerce settings, and that alternative models are needed that minimise the cost to the user, while maintaining recommendation quality.

1 Introduction

Online recommender systems serve as virtual sales assistants, that interact with the user in order to ascertain her product preferences and so direct her to the right product. Essentially they attempt to reproduce the real-world customer-buying processes within the e-commerce domain. A number of models have been proposed to describe the real-world customer-buying process [1,2,3]. All of these can be generalised as follows: first of all, a vague description of the customer's required product is given; the customer might tell the sales assistant specific details about the item they are looking for (e.g. "I'm looking for a nice evening dress for under $250"). Following this, a cyclic interaction between the customer and the sales assistant takes place, where the assistant typically alternates between asking questions and proposing suggestions to the customer until the customer is satisfied or all possibilities have been presented [4].

Currently, the vast majority of research in recommender systems has made what appears to be a legitimate assumption, namely that what works in the real world, also makes sense in the virtual world. We believe that this approach *may* be flawed for two reasons. First of all, a user may not be in a position to answer direct questions about a particular product, either because they do not have a detailed understanding of the product space (a first time PC buyer for example) or because their understanding of the product space differs from the sales assistant's. Secondly, even if the user is familiar with the product space in question, she may be less tolerant of being asked lots of detailed questions by

* The support of the Informatics Research Initiative of Enterprise Ireland is gratefully acknowledged.

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 209–214, 2002.

the online recommender system. Communicating information back to an online assistant takes more effort; the user interface may not be natural (especially for the novice) and the user may have to negotiate a series of menus, selections, and clicks in order to specify even a single product detail. The bottom line is that the more costly a given interaction model is, in terms of user effort, the less likely a user is to engage, and ultimately purchase. And because, in the online world, a user is no more than a click away from the competition, it is very important for e-tailers to make the right decisions when it comes to defining the online experience, and this includes recommender system design.

In this paper we describe our *comparison-based recommendation* approach, which avoids engaging the user in a lengthy dialogue in favour of a more relaxed conversational strategy. Instead of asking the user to provide information about specific product features, they are simply asked to choose one of the recommended items as a preference, and we attempt to learn detailed feature preferences from this feedback. This low-cost form of user interaction has been largely ignored in recommender systems to date. However, our evaluations to date show that this approach allows us to effectively refine customer preferences in order to make high-quality recommendations.

2 Comparison-Based Recommendation

The comparison-based recommendation algorithm consists of 3 main steps: (1) a set of k items are *recommended* to the user that closely match the current query; (2) the user *reviews* the recommendations and indicates a preference case which best matches her needs (*positive* feedback); (3) information about the difference between the selected case and the alternatives is used to *revise* the query for the next cycle. The recommendation process terminates either when the user is presented with a suitable case or when they give up.

2.1 Query Revision

The key to the success of comparison-based recommendation is the way in which the current query is revised to reflect the user's feedback. The ultimate objective is to update query features with features from the preferred case that best reflect the user's implicit preferences. In previous work [5] we have evaluated a number of different query revision strategies. In this paper we will focus on two in particular: MLT (More Like This) and wMLT (Weighted More Like This).

The simplest form of query revision (MLT) is to take each feature of the preferred case as a new query feature. The advantage is that a partial user query can be instantly extended to include additional features (from the preferred case) in the hope of focusing more effectively on the right set of cases during the next recommendation cycle. However, not every feature of the preferred case may be actively preferred by the user; for instance, the user may have preferred a PC because of its low *price* and not because of its low *processor speed*. Because of this we can expect the MLT update strategy to suffer from a form of over-fitting to user

Features	C1	C2	C3	C4
Manufacturer	Dell	Sony	Compaq	Fujitsu
Memory	256	128	128	256
Monitor	15	14	15	12
Type	Laptop	Laptop	Laptop	Laptop
Processor	Celeron	Pentium	Pentium	Celeron
Speed	700	600	700	650
Disk	20	20	40	20
Price	1300	999	1200	1000

Fig. 1. A typical recommendation cycle (PC domain)

feedback and as a result we may find that the user is guided to irrelevant parts of the recommendation space from their early preferences. This can result in long recommendation cycles and the need to examine more cases than necessary.

The wMLT strategy attempts to weight the new query features based on the degree of confidence that we can attribute to them as preferences for the user. The basic idea is that the more alternatives that have been presented to the user, the more confident we can be about learning their preference. For example, in Fig. 1 we can be confident in the user's *Compaq* preference because there is a wide range of alternatives for the *manufacturer* feature. Compare this to the *processor type* feature, where only two alternatives are presented, *Celeron* and *Pentium*; we can not be sure whether the user is genuinely interested in *Pentium* or whether they are simply more interested in *Pentium* than *Celeron*. Therefore, the wMLT strategy transfers all preferred features to the new query but weights them according to the formula shown in Eq. (1), which gives preference to diverse features among the recommended set. For example, the preference for *Compaq* gets a weighting of 1 because all 3 of the rejected cases, R, have unique *manufacturer* values that are different from *Compaq*. In contrast, the preference for *Pentium* gets a weight of 1/3 since the 3 rejected cases only account for one alternative to *Pentium*.

$$weight(f_i, R) = \frac{\#\ of\ alternatives\ to\ f_i\ in\ R}{|R|} \tag{1}$$

3 Evaluation

The following experiments are carried out using a case-base of 120 cases, each describing a unique PC. We compare 2 different versions of our comparison-based recommendation technique, each employing a different revision strategy (MLT, wMLT). Here the MLT version serves as a benchmark against which to judge the performance of the more sophisticated wMLT version. In addition, we implemented a simple similarity-based recommender (SIM) as a further comparison. We use a leave-one-out evaluation methodology, varying values of k (the number of cases returned during each recommendation cycle) from 2 to 5, and q (the number of initial features per query) from 1 to 5. Each case of the case-base

is temporarily removed and used in two ways. First each case serves as the basis for a set of random queries of varying sizes (ranging from 1 to 5 features per query). Second, we select that case in the case-base which is most similar to the current target case. These *best cases* serve as the recommendation targets during the experiments. In total a set of 1200 separate queries are generated and all results are averaged as appropriate.

3.1 Recommendation Efficiency

A key success criterion for any navigation by proposing recommender system is the number of individual cases that the user needs to examine before they settle on a *best case*. In this experiment we note the average number of cases that the user must view before the best case is located. In the case of SIM this is the average position of the best case in the similarity ordered list produced from the initial queries. For the comparison-based recommenders it is the total number of cases presented in each recommendation cycle prior to the final cycle plus the position of the best case in the final cycle. For example, for a $k = 3$ trial where 3 recommendation cycles are needed and the best case is the second case in the final cycle, then the user has needed to examine $(3 * 2) + 2 = 8$ cases.

The results are shown in Fig. 2. Figure 2a indicates that the comparison-based recommenders work well relative to the SIM even though they operate on the basis of minimal feedback from the user during each recommendation cycle. Essentially, the wMLT method requires the user to look at nearly 30%

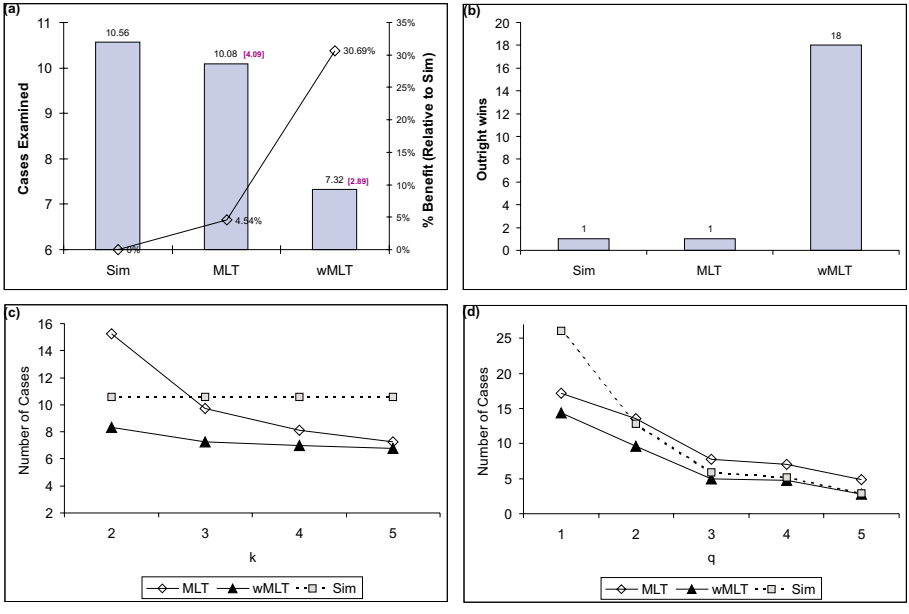


Fig. 2. Recommendation efficiency results

fewer cases than the MLT benchmark. Figure 2c indicates that as k increases the average number of cases examined by the user decreases (of course the SIM plot remains flat here as it does not depend on k). We find that the wMLT technique performs best of all for the different values of k, and is in fact quite insensitive to k, suggesting that the weighting scheme employed by this scheme has a positive effect independent of window-size. Figure 2d shows the average results (over the different k values) for different values of q, the initial query size. This time SIM is affected, with significant reductions in the number of cases that must be examined as q increases, as expected. Once again MLT performs poorly here, losing to SIM for values of $q > 2$. The best overall performance is given by wMLT, which consistently out-performs MLT and SIM, for all values of q.

Analysing the average case results does not tell us if one technique is consistently performing better than another. In fact, a good average could be the result of a single or small number of good trials for an otherwise poorly performing strategy. For this reason, Fig. 2b looks at the number of trials where each technique wins. The graph indicates that the wMLT method is the best performer overall, beating SIM and MLT 90% of the time, that is, in 18 out of 20 of the trials. In other words, even though on the face of it, MLT exhibits a performance that is comparable to wMLT (at least in some trials), MLT (and SIM) rarely perform best overall – wMLT is consistently the outright winner.

3.2 Recommendation Quality

In general the more cases a user needs to examine, the more likely they are to abandon their current recommendation session, leading to a recommendation failure. One way to measure recommendation quality is to look at the number of recommendation sessions that fail to complete within a given number of cases (*failure-rate*). In the following experiment we examine the recommendation failure-rate of the three different recommendation strategies by processing the results of the previous trials in order to compute the number of sessions that terminate (with the best case) within a given threshold number of cases. We focus on thresholds up to 30 cases.

Figure 3a plots the results for each recommendation strategy in terms of the percentage failure rate against the case threshold. The failure-rate for all three algorithms is poor at very low thresholds, as expected. However, the failure-rate decreases rapidly as the threshold increases, and the rate of descent varies depending on the recommendation strategy. Once again the wMLT strategy outperforms MLT and SIM. For instance, for thresholds of up to 10 cases, wMLT has a failure rate of 22% whereas the equivalent failure-rate is 29% and 35% for MLT and SIM, respectively. In other words, MLT has a failure-rate at this threshold (10 cases) that is 1.3 times that of wMLT, and SIM has a failure-rate that is 1.6 times that of wMLT. These relationships can be seen more clearly in Fig. 3b, which plots the failure-rate of MLT and SIM relative to wMLT against the threshold; a relative failure-rate of 2 for SIM at a given threshold, indicates that SIM has twice as many recommendation failures as wMLT at this threshold. Figure 3b clearly shows how MLT and SIM perform poorly compared to wMLT

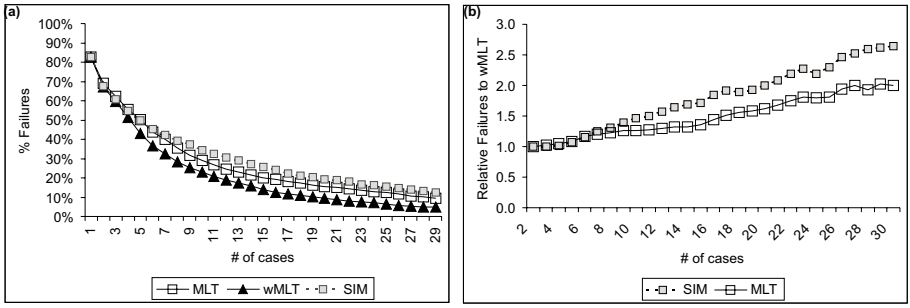


Fig. 3. Recommendation failure-rates

as the threshold number of cases increases; in fact, MLT reaches 1.9 times the failure-rate of wMLT, and SIM reaches 2.6 times the failure-rate of wMLT, at the 30-case threshold. In summary then, the wMLT strategy presents with significantly lower recommendation failure-rates than either of the MLT or SIM strategies.

4 Conclusions

Anecdotal evidence, at least, suggests that while real-world customer-buying models *can* be translated for online e-commerce scenarios, perhaps they should not be (because of the effort these models impose on the user). In our work we have developed the *comparison-based recommendation* approach, which avoids engaging the user in a lengthy dialogue in favour of a more relaxed conversational strategy. Our evaluations to date show promising results for the comparison-based recommendation approach, in terms of its effectiveness and efficiency characteristics. Future work will focus on extending our evaluation into additional recommendation domains and into looking at new revision strategies based on alternative weighting models.

References

1. Bettman, J.: An Information Processing Theory to Consumer Choice. Addison-Wessley, (1979)
2. Howard, J., Sheth, J.N.: The Theory of Buyer Behavior. Wiley, (1994)
3. Shigeta, T.: 30 Lessons on how to be an Excellent Salesclerk. Keirin Shobo, (1995). (in Japanese) Tokyo
4. Shimazu, H.: ExpertClerk : Navigating Shoppers' Buying Process with the Combination of Asking and Proposing. In Nebel, B. editor, Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, pages Volume 2, pages 1443–1448. Morgan Kaufmann, (2001). Seattle, USA
5. McGinty, L., Smyth, B.: Comparison-Based Recommendation. The European Conference on Case-Based Reasoning. Springer, (2002). Aberdeen, Scotland

Design of a Musical Instrument Classifier System Based on Mel Scaled Cepstral Coefficient Supervectors and a Supervised Two-Layer Feedforward Neural Network

Declan McGrath

Dept. of Information Technology, NUI Galway
declan.mcgrath@geminga.nuigalway.ie

Abstract. A Content Based Audio Retrieval system is presented, which uses Mel Scaled Frequency Cepstral Coefficients (MFCCs) as frequency based feature vectors to distinguish between different musical instruments via a Neural Network. Supervectors were formed from the concatenation of 14 dimensional feature vectors (12 MFCCs, 1 spectral centroid and 1 frame energy feature) taken from 213 time slices, over 4 second samples. These feature sets were then used to train a two-layer, feedforward neural network, to segregate instrument samples into categories based on their audio content. The motivation for this work is to create a system, which when presented with an unknown sample, can categorise it as belonging to one of a set of predefined classes. The system was tested on the recognition of instruments given 4 second monophonic phrases. An overall recognition accuracy of 40% was obtained, with drum and bass classes being the best categorised.

1 Introduction

The requirement for a system which can retrieve audio automatically and accurately is apparent. Over the past number of years there has been sharp growth in the amount of sound samples available via online repositories. Manually annotating such volumes of audio material by hand presents a monumental challenge and human subjectivity clouds the definition of suitable categories. Hence, a computer-based solution is an attractive proposition. The greatest difficulty in building such a system lies in the classification step – the manner in which the machine goes about assigning a meaningful identifying label to a sound sample. The musical instrument recognition system designed can be broken down into two subsystems, the front end feature extractor and the back end classifier. The front end consists of a series of processes which convert the raw audio data to Mel Scaled Cepstral Coefficient based features. These characterise a sample based on the actual audio content of its waveform. The output of this process is then passed to a neural network which categorises the sample and is known as the system back end. An overview of the entire system is presented in Fig. 1 [1]. The front and back ends of the system shall be examined separately.

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 215–220, 2002.

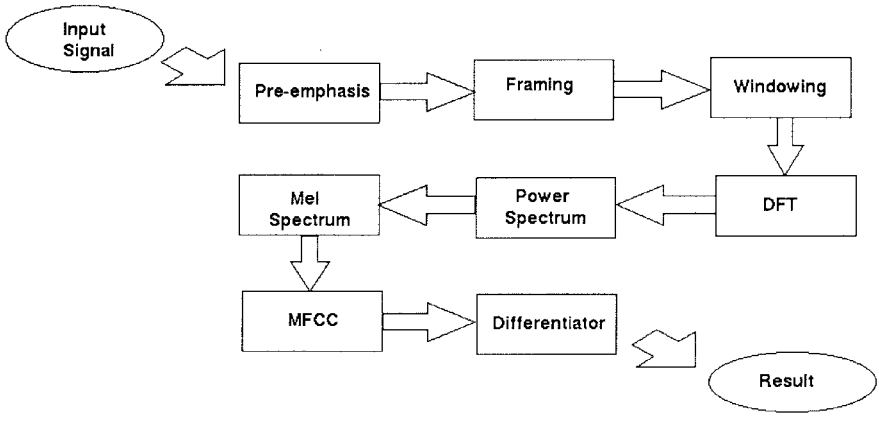


Fig. 1. Musical instrument classifier system overview

2 System Front End: Feature Extraction

Mel Scaled Cepstral Coefficients [2] were chosen as the underlying feature set for this system due to their ability to model the human auditory system and their compact representation of data. Auxiliary features incorporated into the system include the Spectral Centroid and Amplitude Envelope characteristics [3,4]. The extraction of MFCC features from an audio waveform is a multistage process, which shall now be outlined.

A set of wave files obtained from internet databases [5,6] were used as a source of audio data. Each audio waveform was pre-emphasised by flattening the sound spectrum and carrying out normalisation [1]. For the purposes of the accurately generating MFCCs, a stationary signal (one whose statistics do not change over time) is required. Hence, frames of 25 ms duration were extracted from the overall waveform, with an overlap of 25% between adjacent frames. This process is known as frame blocking (see Fig. 2) and further processing is carried out on a per frame basis. Firstly, a Hamming window is applied to the frame in order to provide the best possible frequency representation. A Discrete Fourier Transform (DFT) is then employed in order to convert the windowed frame from the time domain to the frequency domain. The power spectrum of the DFT is obtained in order to resolve the real and imaginary components generated by the DFT [4]. Linear to mel scale conversion is carried out through the application of a mel scaled filterbank to the power spectrum; the resulting coefficients being more relevant to human perception of sounds. The filterbank itself consists of a fixed number of triangular filters evenly spaced across the mel-frequency scale with heights scaled to unity [4] and is defined by

$$F[l] = \sum_{k=0}^{N/2} S[k] M_l[k] \tag{1}$$

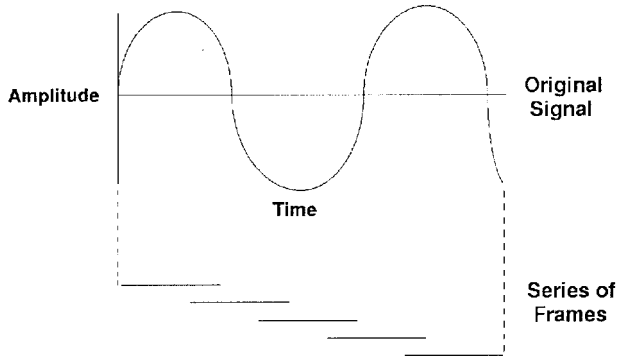


Fig. 2. Overview of the frame blocking process

where M_l denotes a given mel filter, N = length of the DFT, $S[k]$ = power spectrum value at coefficient k and $F[l]$ = filterbank output for filter l. The filterbank is applied to the power spectrum of the audio sample prior to the application of an inverse Fourier transform (IDFT) or more efficiently a Discrete Cosine Transform (DCT). The IDFT or DCT decorrelates the cepstral coefficients, making possible the use of diagonal covariance matrices (Gaussian Models) in the statistical modelling of the feature set. This results in the generation of 26 MFCC values, called the mel-cepstrum, which describe the spectral shape in the Quefrency domain. The most important MFCC components are the lower order cepstral coefficients which model the overall spectral shape. The least important components are the higher order coefficients which model the detail of the fine spectral structure. The first 12 MFCCs, excluding the zeroth coefficient, are retained [4].

Auxiliary features were incorporated to improve system performance – the spectral centroid and the frame energy. The spectral centroid gives a measure of the overall brightness of a sample and can be defined as the midpoint of the spectral energy distribution of a given sound. It forms a single dimension of the feature vector can be calculated from the equation

$$SC = \frac{\sum_{i=1}^{I} a[i] * f[i]}{\sum_{i=1}^{I} a[i]} \tag{2}$$

where I denotes the maximum discrete frequency bin, $a[i]$ = amplitude value for discrete frequency bin i and $f[i]$ = centre frequency value for discrete frequency bin i [3]. The frame energy is equivalent to the sum of the magnitudes of a given frame's amplitudes in the time domain. It is calculated on the frame prior to windowing and is defined by

$$FE = \sum_{n=1}^{N} |s[n]| \tag{3}$$

where FE = Frame Energy, $s[\ ]$ denotes a single frame and N = number of elements in frame. The frame energy is normalised according to the maximum frame energy of the current sample [4] before forming the final dimension of the feature vector. The effect of the front end thus far, was to produce a 14 dimensional feature vector for each frame. In total, 213 such vectors were generated per sample. The dimensions of all feature vectors were converted to binary representation and concatenated together to form a supervector. The position of each feature vector within the supervector is related to the temporal position of the frame within the original signal. Such concatenation had been suggested by Foote [7], who proposed that such techniques could be used to model time variation. The supervector served as an input to the neural network.

3 System Backend: Neural Network Classifier

The Stuttgart Neural Network Simulator [8] was chosen as a backend for the system. A fully connected, feedforward neural network was used to process the feature vectors extracted by the front end. A two-layer architecture was decided upon so that a hidden layer could be used to extrapolate higher order statistics from the underlying data. Training of the network was carried out via supervised learning, as opposed to unsupervised, so that predefined, distinct categories could be used as opposed to the network generating its own. The Stuttgart Neural Network's Analyze tool was used to analyse the networks output with respect to overall results and individual class results. The Analyze tool can be invoked in different modes, which vary according to how one wishes the results to be interpreted. For this system, the WTA (Winner Takes All) mode was used. This meant that the highest ranking value in the neural network output was determined to the class verdict.

4 Experimental Work and Results Analysis

The system was tested on 4 second samples taken from 65 wave files comprising 12 classes of instruments, both monophonic and polyphonic. Each instrument class contained multiple instrument examples and samples taken from different sessions in order to make conditions as realistic as possible. Two experimental runs were carried out. Results of the first experimental run are presented in Table 1.

On analysis of the results, bass (42.86%) and drum (87.50%) samples were found to be the best categorised, compared with an 8.3% random probability of the network identifying a class by chance. Despite the heterogeneity of the drum category, the system still identified almost all unknowns. The drum and bass classes were the most trained and tested of all the classes indicating that, in general, the more examples presented the higher the recognition rate.

Piano samples were categorised with an accuracy of 50%. This figure would probably have been even higher but for the fact that some piano samples were

Table 1. Individual class results for test sets used in experimental work

	Run 1		Run 2	
Class	**Accuracy**	**No. of Test Patterns**	**Accuracy**	**No. of Test Patterns**
Brass	0%	2	50%	6
Choir	100%	1	0%	1
Guitar-Flute	0%	1	0%	1
Organ-Violin-Piano-Soprano	0%	2	0%	2
Organ-Brass	100%	1	0%	1
Piano	50%	2	0%	3
Piano-Cello	0%	2	0%	2
Bass	42.86%	7	42.86%	7
Bell	0%	2	0%	2
Drum	87.50%	8	87.50%	8
Keyboard	0%	3	33%	3
Strings	0%	4	25%	4
Overall	**40%**	**35**	**40%**	**40**

contaminated with additional instrument influences, making their categorisation more difficult. When the raw network output was examined, it appeared that the system tended to confuse a given polyphonic category with other polyphonic categories. This is likely due to the fact that the composite nature of such classes made for a "bigger" sound than monophonic classes, which would be picked up by the frame energy feature vector in particular. The choir and organ-brass classes both returned recognition accuracies of 100%. Caution must be exercised when interpreting such results due to the small size of the test sets used, although there was some similarity between the training and test sets of such instruments. Although many of the other categories achieved low recognition rates, an examination of the raw network outputs, in many cases revealed that the network would have ranked the correct category second, just behind the category returned as the result. Fine tuning of the feature set may allow such instruments to be better categorised, a candidate for future work. In the case of some instruments, insufficient number of training and/or test samples were used.

A major shortcoming of the first run was that the brass test was too small and dissimilar from the training set. Hence, a second experimental run was carried out, in which brass training and test sets were adjusted accordingly. The results obtained are shown in Table 1. The results of both runs are similar demonstrating experimental repeatability. The major difference is that over a test set of 6 brass samples, a recognition rate of 50% was obtained. The system has become more specialised at recognising brass samples at the expense of choir, guitar-flute, piano and organ-violin-piano-soprano samples. However, if more test samples were available for each instrument it would be likely that the system would reflect a better performance for such classes. It can be seen that rectifying

problems and confusions concerning recognition within a particular category can resolve (or create) problems in another category. Keyboard and brass samples were previously confused drastically in the first run. In the second run, the network is clearer on the definition of the brass category and thus is better able to differentiate it from the keyboard category. These observations illustrate that it is always of the utmost importance to maintain a global perspective when dealing with neural networks. Other classes have a major bearing on the effectiveness of the network when it is asked to categorise a given sample.

5 Conclusion

The system has shown itself to be capable of segregating instruments into different classes through use of the MFCC-based supervector feature. The performance of the system was best on higher quality, monophonic samples such as drum and bass, as opposed to composite mixtures. With increased training, it was also capable of dealing with lower quality samples such as the brass samples used in these experiments. The system was particularly effective at distinguishing short single note samples such as drum and certain bass samples, due to the temporal nature of the supervector. Future work shall focus on testing the system on a much larger, higher quality dataset as well as comparing the performance of the system with that of human subjects. Additionally, the system may be modified to operate on excerpts of songs and the feature set fine tuned.

References

1. Seltzer, M.: SPHINX III Signal Processing System. CMU Speech Group (1999)
2. Davis, S. B., Mermelstein, P.: Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. In: IEEE Trans. ASSP August (1982) 357–366
3. Sandell, G.: SHARC Timbre Database. Parmly Hearing Institute, Loyola University Chicago (1995) Web ref: `http://sparky.parmly.luc.edu/sharc`, Last accessed: 2/04/2002
4. Zheng, F., Zhang, G. L.: Integrating the Energy Information into MFCC. In: Proc. Int. Conf. on Spoken Language Processing, October 16-20, Vol.1 (2000) 389–292
5. Green, M., Nelson, P., Tchou, C., Watson, R., Hedberg, M.: The Mod Archive (2002) Web Ref: `http://www.modarchive.com/waveworld`, Last Accessed: 11/04/2002
6. Potulski, F.: Freesamples.de (2002) Web Ref: `http://www.freesamples.de`, Last Accessed: 11/04/02
7. Foote, J. T.: Content-Based Retrieval of Music and Audio. In: C.-C. J. Kuo et al. (eds.) Multimedia Storage and Archiving Systems II, Proc. of SPIE, Vol. 3229 (1997) 138–147
8. Zell, A.: Stuttgart Neural Network Simulator, User Manual V4.2. University of Stuttgart (2000) Web ref: `http://www-ra.informatik.uni-tuebingen.de/SNNS`, Last accessed: 25/03/2002

An Interactive Learning Environment for Knowledge Engineering

David McSherry

School of Information and Software Engineering
University of Ulster, Coleraine BT52 1SA, Northern Ireland
dmg.mcsherry@ulst.ac.uk

Abstract. While the importance of verification and validation (V&V) techniques in knowledge-based systems (KBS) development is well recognised, little attention (if any) has been paid to their potential role in the context of artificial intelligence teaching. We present an interactive learning environment that aims to promote independent learning by students undertaking knowledge-engineering assignments. An innovative feature of the approach is the use of V&V techniques to provide automated feedback to students prior to submission of coursework.

1 Introduction

In our view, the best way to develop knowledge-engineering skills is through practical experience in the development of KBS applications. However, errors such as incompleteness and inconsistency are common in knowledge bases (KBs) constructed by novices. Such errors can be very difficult even for an experienced practitioner to detect by manual inspection. Not only is this likely to impede progress in the achievement of learning outcomes, it also creates an obvious problem for coursework designers faced with the assessment of large numbers of students.

We present an interactive learning environment for knowledge engineering and describe how it has helped to enable students undertake knowledge-engineering assignments that might otherwise be too difficult, or impossible to assess. An innovative feature of the approach is the use of V&V techniques [1,2] to provide automated feedback to students prior to submission of coursework. The environment combines the usual components of an expert system shell (inference engine, working memory, and KB editor) with a KB *verifier* that the student can use at any stage to obtain feedback in the form of error reports and warnings.

In Sect. 2, we describe the expert system component of the learning environment and a typical knowledge-engineering assignment involving the creation of a KB for the expert system. In Sect. 3, we demonstrate the ability of the KB verifier to detect common errors such as missing rules and provide automated feedback to students. Our conclusions are presented in Sect. 4.

2 The Expert System

Implemented in Prolog, the expert system component of the learning environment is a basic rule-based expert system with a backward-chaining inference mechanism. At any stage in the construction of a KB, the student can present test cases to the expert sys

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 221-227, 2002.

tem to check that the conclusions it reaches are correct. The system's ability to explain its reasoning on request can play an important role in the validation process. For example, an explanation of how an incorrect conclusion was reached may help the student, in her role as knowledge engineer, to identify the rule that is causing the error and the corrective action required.

2.1 Example Assignment

In a typical knowledge-engineering assignment, students are provided with a decision tree for a diagnosis or classification task (e.g. fault finding in a computer system) and required to:

- Use the decision tree to create an initial KB for the expert system
- Identify the *goals* that the expert system will attempt to prove
- Extend the KB to include rules for appropriate *sub-goals* in the domain
- Provide *questions* to be used by the expert system when requesting data from the user
- Test, document, and demonstrate the prototype expert system

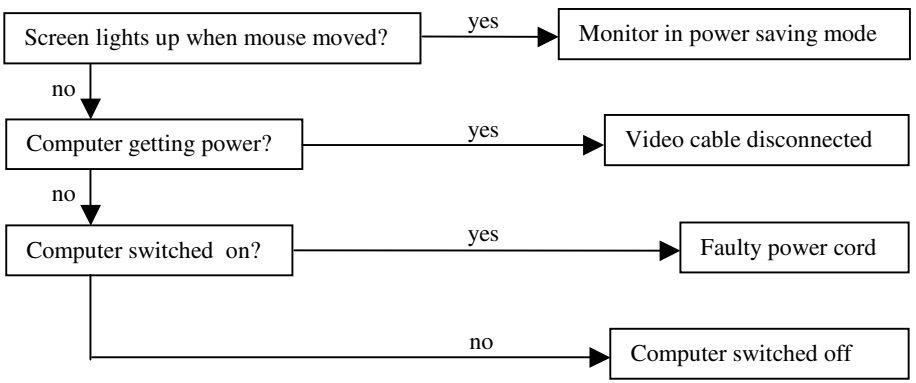


Fig. 1. Simplified version of a decision tree for diagnosis of the fault "screen is dark" in a computer system

Though lacking the modularity often seen as an advantage of rules [3], decision trees are often useful as an *intermediate* method of knowledge representation in expert systems development [4]. Knowledge captured in a decision tree, for example by a machine learning algorithm, can easily be transformed into rules. The attribute values on each path from the root node to a leaf node provide the conditions of a rule with the outcome class at the leaf node as its conclusion. Figure 1 shows a highly simplified version of a decision tree for fault diagnosis in a computer system that has been used in knowledge-engineering assignments. The following is one of the 4 rules that can be derived from the example decision tree.

if screen lights up when mouse moved = false
and computer getting power = true
then video cable disconnected = true

A set of rules derived in this way from a decision tree has two important properties. First, the rules are *mutually exclusive*; that is, no combination of values of the attributes in the tree can exist which satisfies the conditions of more than one of the rules. Second, the set of rules is *complete* in the sense that every combination of values of the attributes in the decision tree is covered by one of the rules. On the other hand, it is possible that rules derived from a decision tree may include redundant conditions; that is, the rules may not be *maximally general* [3,4]. A disadvantage of decision trees in comparison with rules is their inability to represent *sub-goals*; that is, intermediate hypotheses that can help to reduce the problem-solving process to simple steps. However, when a decision tree is transformed into rules, conditions in the decision tree such as 'computer getting power = true' can be treated as sub-goals in the resulting expert system [4].

2.2 Example Knowledge Base

Figure 2 shows a KB constructed from the example decision tree. In addition to the rules derived from the decision tree, there are rules for proving the sub-goals 'computer getting power = true' and 'computer getting power = false'. The goals that the expert system will attempt to prove are the conclusions at the leaf nodes of the decision tree. Questions to be used by the expert system when requesting data from the user have also been provided.

Rule 1 **if** screen lights up when mouse moved = true **then** monitor in power saving mode = true

Rule 2 **if** screen lights up when mouse moved = false **and** computer getting power = true **then** video cable disconnected = true

Rule 3 **if** screen lights up when mouse moved = false **and** computer getting power = false **and** computer switched on = true **then** faulty power cord = true

Rule 4 **if** screen lights up when mouse moved = false **and** computer getting power = false **and** computer switched on = false **then** computer switched off = true

Rule 5 **if** power light on = true **then** computer getting power = true

Rule 6 **if** fan can be heard = true **then** computer getting power = true

Rule 7 **if** power light on = false **and** fan can be heard = false **then** computer getting power = false

Goals: monitor in power saving mode, video cable disconnected, faulty power cord, computer switched off

Questions:

question('screen lights up when mouse moved', 'Does the screen light up when the mouse is moved?')
question('power light on', 'Is the power light on?')
question('fan can be heard', 'Can the fan be heard?')
question('computer switched on', 'Is the computer switched on?')

Fig. 2. KB constructed from the example decision tree

```
ES:   Does the screen light up when the mouse is moved?
User: no
ES:   Is the power light on?
User: why
ES:   Because if power light on = true
      then computer getting power = true
ES:   Is the power light on?
User: why
ES:   Because if screen lights up when mouse moved = false
      and computer getting power = true
      then video cable disconnected = true
ES:   Is the power light on?
User: unknown
ES:   Can the fan be heard?
User: yes
ES:   Possible conclusion: video cable disconnected
```

Fig. 3. An example consultation with the expert system

Figure 3 shows an example consultation with the expert system showing its ability to explain its reasoning on request. At the end of a consultation, the user can also ask how the conclusion was reached, or how any sub-goal was proved. The example consultation also shows how the expert system can sometimes reach a conclusion when relevant data is not available. For example, when the user is unable to say if the power light is on (Rule 5), the system uses Rule 6 instead to prove that the computer is getting power.

3 The Knowledge Base Verifier

We now describe the KB verifier, as currently implemented, that the student can use at any stage to obtain feedback in the form of error reports and warnings.

3.1 Circular Reasoning

To illustrate the ability of the KB verifier to detect circular reasoning, consider the effect of adding the following rule to the example KB.

if computer switched off = true **then** power light on = false

Though obviously correct, this rule is an example of *deductive* reasoning whereas expert systems tend to rely on *abductive* reasoning from observed symptoms to their possible causes. When both types of rule are combined, the result may be circular

reasoning. Figure 4 shows the error reported by the KB verifier when applied to the modified KB.

Circular reasoning detected in the following rules:

if computer switched off = true **then** power light on = false

if power light on = false **and** fan can be heard = false **then** computer getting power = false

if screen lights up when mouse moved = false **and** computer getting power = false **and** computer switched on = false **then** computer switched off = true

---

Fig. 4. Detection of circular reasoning

3.2 Missing Rules

To illustrate the ability of the KB verifier to detect missing rules, Fig. 5 shows the error message reported when Rule 7 is deleted from the example KB. The KB verifier similarly checks that there is a matching rule for every top-level goal.

There is no rule or question for computer getting power = false in the rule:

if screen lights up when mouse moved = false **and** computer getting power = false **and** computer switched on = true **then** faulty power cord = true

---

Fig. 5. Identifying gaps in the knowledge base

3.3 Missing Questions

A common error that can be difficult to detect by manual inspection is inconsistency in the wording or spelling of rule conditions and question declarations. For example, consider the effect of replacing the first question declaration in the example KB by the slightly different version:

question('screen light up when mouse moved', 'Does the screen light up when the mouse is moved?')

If the student attempts to run the expert system, it will immediately report that it is unable to reach a conclusion. The reason is that it is unable to find a question (or rule) in the KB that matches the first conditions of Rules 1-4. The student can easily identify the cause of the problem with the help of the KB verifier. The error report it pro-

duces in this case is similar to the example shown in Fig. 5. The KB verifier also checks that the conclusion of every rule is either a condition in another rule, in which case it is assumed to be a sub-goal, or is declared as a top-level goal.

3.4 Missing Rule Conditions

Students often omit conditions from the rules they derive from decision trees, perhaps believing them to be redundant. For example, the following rule obtained by deleting the first two conditions of Rule 4 is certainly correct:

if computer switched on = false **then** computer switched off = true

However, Rules 1-4 are no longer mutually exclusive, which means that it may be possible for the expert system to reach more than one conclusion from a given set of data. Depending on the order in which goals are listed in the KB, the conclusion it reaches may differ from the one given by the decision tree. Even worse, the two conclusions may be mutually exclusive. Figure 6 shows the warning given by the KB verifier when applied to the modified KB.

Two different goals (monitor in power saving mode = true and computer switched off = true) can be proved with the following data:

computer switched on = false, screen lights up when mouse moved = true

---

Fig. 6. A warning that more than one conclusion can be reached from the same data

4 Conclusions

We have presented an interactive learning environment for knowledge engineering that aims to promote independent learning by providing automated feedback to students prior to submission of coursework. The system has been successfully used in artificial intelligence teaching and helped to achieve our objective of enabling students to undertake knowledge-engineering assignments that might otherwise be too difficult, or impossible to assess. While many students have reported that they found the system helpful and easy to use, the benefits are difficult to quantify. It would of course be interesting to compare the results produced with and without the assistance of the KB verifier, but such a controlled experiment would be difficult to justify in view of the likelihood that some students would have an unfair advantage.

References

1. Gonzalez, A., Dankel, D.: The Engineering of Knowledge-Based Systems: Theory and Practice. Prentice-Hall, Englewood Cliffs, New Jersey (1993)

2. Preece, A.: Building the Right System Right: Evaluating V&V Methods in Knowledge Engineering. Expert Update **1** (1998) 26-34
3. Cendrowska, J.: PRISM: an Algorithm for Inducing Modular Rules. International Journal of Man-Machine Studies **27** (1987) 349-370
4. McSherry, D., Fullerton, K.: Intermediate Representation of Knowledge in the Development of a Medical Expert System. Biomedical Measurement Informatics and Control **1** (1986) 75-82

Customising a Copying-Identifier for Biomedical Science Student Reports: Comparing Simple and Smart Analyses

Julia Medori[1], Eric Atwell[1], Paul Gent[2], and Clive Souter[3]

[1] School of Computing, University of Leeds, Leeds LS2 9JT, England
{medorij,eric}@comp.leeds.ac.uk
http://www.comp.leeds.ac.uk/nlp/
[2] School of Biomedical Sciences, University of Leeds, Leeds LS2 9JT, England
J.P.Gent@leeds.ac.uk
[3] Centre for Joint Honours in Science, University of Leeds, Leeds LS2 9JT, England
D.C.Souter@leeds.ac.uk

Abstract. The aim of our project is to develop a system for detecting student copying in Biomedical Science laboratory practical reports. We compare contrasting approaches: "simple" methods Zipping, based on a standard file-compression tool, and Bigrams, a basic comparison of frequent bigrams; and "smart" methods using commercial-strength plagiarism-checking systems Turnitin, Copycatch, and Copyfind. Both approaches successfully flag examples of copying in our Test Corpus of 218 student courseworks, but Copycatch provides more user-friendly batch-processing mechanism. Human experts go beyond word-based pattern matching, and take account of knowledge specific to our domain: methods and questions can legitimately be copied, whereas originality is more important in the Discussion section.

1 Introduction

The aim of our project is to develop a system for detecting student copying in laboratory practical reports, customised to a specific genre/subject, in our case initially Biomedical Science first-year reports. We are aware of many commercial plagiarism detectors, but Biomedical Science teaching staff believe these generic systems are generally too sophisticated or complicated for this specific problem.

We have collated a Test Corpus of example student reports in the specific genre, to identify characteristic features. We have also interviewed Biomedical Science teaching staff to elicit/confirm significant diagnostic features which can identify copying, and identify the level of copying considered unacceptable (as some overlap in lab reports is expected).

Unlike existing surveys of generic plagiarism detection systems, we have tried to evaluate systems against the detailed specific requirements identified above. A number of candidate systems were considered, including:

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 228–233, 2002.

- CheatChecker [1] – an n-gram based tool
- SIM [2] and YAP [3] based on longest common sub-sequence approach.
- Detecting "unusual" similarities between a closed set of essays (or programs) is called collusion and the basis for Malcolm Coulthard and David Woolls' Copycatch tool [4]
- Copyfind [5] is an example of a system aimed at scientific rather than humanities documents
- Turnitin is probably the most widely-known system, but the website [6] gives scant details of the underlying methods
- Clough [7] is investigating more sophisticated Natural Language Processing techniques.

2 Test Corpus of Biomedical Science Student Laboratory Reports

Evaluation using a test corpus is a well-established methodology in Natural Language Processing research, and is appropriate to this task: in our case, the aim is to test candidate systems and compare detection (and "red-herring") rates. The initial test corpus used in our study was composed of 103 Biomedical Student reports. As there turned out to be few examples of 'real' plagiarism in our initial corpus, we extended it by adding 94 reports written by second year students, and by artificially generating 19 partly-copied documents from the original reports. The first year assignment described the experiment and gave instructions on how to use the computer simulation to generate results; then the students had to produce tables and graphs and answer a few questions to analyse the results. It seemed likely that any plagiarism we would find would be in the text of the answers. The second year students had to report on 3 different experiments. Unlike the first year students reports, these students didn't have to answer specific questions; the text should be more 'free' and show more originality. However, they still follow the same structure: Introduction, Methods, Results, Discussion.

To check the sensitivity of each plagiarism detection software tested, we created 19 new files: each composed of one file from which 5%(then 10%,..., 95%) is removed and replaced by a portion of another file. We also added in our corpus one example of plagiarism that had been detected previously.

A characteristic of this genre is that a lot of the vocabulary (being mostly from the science domains) will be shared between all the reports. The answers being short, not a lot of originality will be noticeable for each student. As most of the plagiarism checkers available at the moment are developed to detect plagiarism either in humanities essays (high level of originality) or in programming (with limited vocabulary), there is no existing copying checker for the specific problem of scientific reports: low level of originality but no limited vocabulary.

Note that the names throughout this report are made-up and are not the real names of the students.

3 Zipping, a Simple Compression-Based Approach

As a baseline of our study, we will test a method inspired by the article [8]. This article suggests that using file compressors or zippers, we would be able to measure a 'distance' between files according to their similarities.

The algorithm of common zippers (like gzip) finds duplicated strings and marks up the matching strings with a pointer; which means that, by appending a questioned text to different corpora (e.g. of different authors or languages) and measuring the difficulty of zipping these files, we can measure the level of similarity between two files. The smaller the zipped file of the concatenation, the more common elements the questioned text shares with the corpus.

4 Turnitin: A Commercial System

The first commercial system we used is Turnitin.com, a web-based plagiarism detection service first designed to detect material cut and pasted from the Internet but also detects similarities between submitted papers; Leeds University has subscribed to the service [10]. The submission of papers is done by means of laborious cut-and-paste of the text from each document. After about 24 hours, an Originality Report TM is sent onto the user's account for each submitted paper.

We found that TurnItIn gives quite good results. However, the fact that it gives an index of similarity between the file and all papers found on the web and in the database leads it to miss a few cases of plagiarism, especially in the case of the pair 'Colin-David'. Turnitin gives as a result the level of similarity between the submitted paper and all the papers in the database and the data found on the web; it seems a waste of time checking on the web and other papers than the ones in the class. A more significant shortcoming is the laborious input method. On the positive side, Turnitin underlines the similarities, making them easier to visualise; and it makes it easier to detect partial plagiarism.

5 CopyCatch

These conclusions lead to the next step of our study, which would be to test CopyCatch, a software developed by Woolls from CFL Software Development. After reading the JISC report [9], it seems that it may be more appropriate to our specific needs as it gives instant results, sorting them by percentage of similarity between 2 files.

The main characteristic of this software is that it is easy to use, especially the submission process: it allows the user to browse and select the files to check. The results are almost immediate. It outputs a list of pairs of files sorted by percentage of match between them. It is then possible to have a list of the vocabulary or phrases shared between the 2 files and it can also mark up the files highlighting the similarities between them. It allows the user to check Word documents as well as text, rtf and HTML files: no previous conversion is needed.

CopyCatch finds without difficulty all our 'real' cases of plagiarism. It even finds the pair 'Colin-David', which was not an obvious case of plagiarism and which we would have missed using TurnItIn. The selection of our files in a few clicks is a very important practical advantage.

6 CopyFind

As another alternative, we decided to test another CopyFind [5], available for Linux as well as for Windows. This software requires that the Copyfind executable and all the files to be checked are in the same folder. The user has to add a list of all the files that have to be checked. These can be Word documents. The user can define a threshold of similarity. The system will then output a file containing all the pairs of files and the number of matched words for each of them. An HTML report will then be created for every pair which are above the threshold, with underlined similarities.

We ran the program with standard parameter values; the results are quite good, but it doesn't detect the pair 'TestFile-TestFile2'. To find this pair in the results, we have to lower the threshold by changing the minimum number of matching words to report from 500 (value suggested) to 300. The modification of this threshold doesn't change any of our other results.

CopyFind is a program that works quite well and gives a clear output underlining the similarities between the files. The results are immediate, but its submission method is cumbersome.

7 Another 'Simple' Approach: Bigrams

This method is a 'home-grown' method based on the idea that when 2 files are similar, they should share a high number of their bigrams. The programs are all implemented in Perl. We tested this method first with character bigrams and then, word bigrams. In the first place, we create a model for each file in our corpus by counting all the occurrences of each character bigrams in the file. To compare two files, we simply see whether the most frequent bigrams are the same in both files. We compare the results for a model containing 10, 20, or 30 most frequent bigrams. The results even for "top 30" character bigrams were disappointing: this method will flag copying of an entire document, but below a total match, most files share a large proportion of character bigrams.

We tried the same experiment with word-bigrams, and got better results: word-bigrams are better discriminators.

8 Result Comparison between the 5 Experiments

Throughout our study, we discovered several occurrences of plagiarism in our corpus; These are listed in Table 1. In this table we show which system was able to point out each case of plagiarism. It is considered as positive (marked with

Table 1. Comparison of results (key: '+' = detected, '-' = not detected)

	pointed out by				
Known cases of copying	Zipping	TurnItIn	CopyCatch	CopyFind	Bigrams
Albert 1-2	+	+	+	+	+
Barbara 1-2	+	+	+	+	+
TestFile - TestFile2	+	+	+	+	+
TestFile - SourceA	-	+	-	-	-
TestFile - SourceB	-	+	-	-	-
TestFile2 - SourceA	-	+	-	-	-
TestFile2 - SourceB	-	+	-	-	-
Geraldine - Fred(ass2)	+	+	+	+	+
Jane - Naomi(ass2)	+	+	+	+	+
Colin - David(ass2)	+	-	+	+	+/-
Jane - Naomi(ass3)	+	+	+	+	+
Constructed plagiarisms	up to 20%	up to 30%	up to 30%	up to 20%	
ScanFile1 - ScanFile2	+	+	+	+	+

Table 2. Usability of the five systems tested: submission and output

	Usability	
systems	submission	output
Zipping	hard-coded into the program	sorted list of pairs by value of 'distance'
TurnItIn	Copy+Paste documents individually	Report with similarities highlighted
CopyCatch	select from file manager window	Statistics+ sorted list of pairs by % of match + files side by side with similarities highlighted
CopyFind	make a list of files	HTML reports for each pairs above threshold with underlined similarities
Bigrams	give directory name	Sorted list of pairs

a '+' in Table 1) if the paper was found at the top of the list of results. In the case of the pair 'Colin-David (Assignment2)' in the word-Bigrams experiment, the pair was only shown within the first seven results after three false-positives. Therefore, it was marked with a '+/-' as it was detected, however, whether this pair of assignments would be manually verified would depend on the amount of time given to the task by the user. If the plagiarism is detected considerably down the list of results (where there is little chance it would be checked by a user manually checking every script in list order) it is marked with a '-' in Table 1.

9 Discussion

In the tests reported above, both approaches successfully flagged examples of "blatant" copying, but results were less clear-cut when reports had small amounts of overlapping text. Results based on our Test Corpus indicate that we need to

take account of knowledge specific to our domain. In Biomedical Science laboratory reports, students must always write four sections: Introduction, Method, Results, Discussion. In fact this is standard practice even for research journal papers; and this format is a mandatory part of the course work specification. Biomedical Science lecturers know that certain parts of the text can legitimately be copied (the given specification of Method, and specific questions set by the lecturer), whereas originality is more important in other parts (notably the Discussion section). Ideally our copying-checker should include some "intelligence", to make it ignore overlap in unimportant sections, and focus on the Discussion section; overlap in Discussion is a much surer indicator of copying.

We have analysed the requirements of our specific problem, through discussion with Biomedical Science teaching staff. We have surveyed a range of candidate systems, and evaluated five systems (Turnitin, Copycatch, Copyfind, and two "home-grown" systems, Zipping and Bigrams) in experiments with a Test Corpus of Biomedical Science student laboratory reports. We concluded that none of the systems stood clearly above the rest in their basic detection ability, but that Copycatch seemed marginally more successful, and more significantly it had the most appropriate user interface for our needs as it ran on a local PC and simplified the analysis of a whole directory of lab reports. Following personal contact with Copycatch developers, we agreed to be the first beta-test site for the enhanced Java version. We installed Copycatch on Biomedical Science computing network for future use by staff. A side-effect of availability and use copy- checking software is the need to migrate from paper-based report submission and processing to electronic submission, probably via student pigeon-holes in the Nathan Bodington virtual building. The School of Biomedical Sciences has decided to require electronic submission of all laboratory reports in future, to facilitate use of copying-detection software. During tests, we found evidence to support suspected cases of copying, and discovered at least one new (previously unknown) case of student copying, which was reported to the appropriate authority.

References

1. CheatChecker: `http://www.cse.ucsc.edu/~elm/Software/CheatChecker/`
2. SIM: `http://www.few.vu.nl/~dick/sim.html`
3. YAP: `http://www.cs.su.oz.au/~michaelw/YAP.html`
4. CopyCatch: `http://www.copycatch.freeserve.co.uk`
5. CopyFind: `http://plagiarism.phys.virginia.edu/software.html`
6. Turnitin: `http://www.turnitin.com/`
7. Clough, P.: `http://www.dcs.shef.ac.uk/~cloughie/`
8. Benedetto, D., Caglioti, E., Loreto, V.: Language Trees and Zipping. Physical Review Letters, Vol. 88,4 (2002)
9. Bull, J., Collins, C., Coughlin, E., Sharp, D.: Technical Review of Plagiarism Detection Software Report. JISC (2001)
10. Wassall, Terry: On-line Plagiarism Detection Projects 2001-2002. FLDU Technical Report, University of Leeds (2002)

A Hybridised GA for the Steiner Minimal Tree Problem

R. Panadero[1] and J.L. Fernández-Villacañas[2]

[1] Departamento de Telemática
[2] Departamento de Teoría de la Señal y Comunicaciones
Universidad Carlos III de Madrid,
Av. de la Universidad 30, 28911 Leganés, Madrid, España
pepe@tsc.uc3m.es, rpa@it.uc3m.es

Abstract. In this paper the application of a Genetic Algorithm (GA) (including operators such as mutation, slicing, gluing and crossover) to solve the Steiner Tree Problem is described. Extra heuristics have also been considered in order to improve the quality of the results as well as the computational time. The results obtained with our algorithm fall close to the optimal solution, and compare positively with those obtained by other authors [7].

1 Introduction

GAs have been applied to different aspects of the Steiner tree problem (e.g. Hesser [3] and Julstrom [4]). Most recently, work has been done in areas related to the Steiner Problem in Graphs, where the GAs are designed in order to find, within a given graph, a minimum subgraph spanning a subset of nodes [1,10].

The Euclidean Steiner tree problem is by far one of the most studied geometric Steiner tree problem variants (the other one is the rectilinear Steiner tree). It consists of finding the shortest interconnection, a Steiner Minimum Tree (SMT), of a set of n terminals with respect to the Euclidean L2-metric. This problem has been shown to be NP-hard [2].

More precisely, the SMT is defined as follows: a number of given nodes (GN hereafter) in the plane can be joined by edges such that all nodes are connected to form a tree. Numbers can be attached to all edges defining their Euclidean lengths. The length of the tree is the sum of the lengths of these edges. Among all possible trees one or more can be found with minimal length. These are called the Minimum Spanning Trees (MST). However, a still shorter tree can be found if additional nodes, so called Steiner nodes (SN hereafter), are introduced. The computational effort to construct the optimal solution is very high [6], thus the construction of SMT is a problem well suited for applying Evolutionary Computation.

The rationale of this paper is as follows: in Sect. 2 we describe the GA in detail. In Sect. 3 results are presented; finally, in Sect. 4, conclusions and future work are drawn.

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 234–239, 2002.

2 The Genetical Algorithm

An evolutionary algorithm has been designed to build the SMT. This algorithm uses some operators such as slicing, gluing, mutation and crossover, as well as a random process for selecting the individuals that compounds each new generation. Each chromosome (or individual) is a vector (of real numbers) of length l. The chromosome consists of a set of x-y coordinates that represent the exact location of each SN. The length of the chromosome, l , is $2*(numberofSteinernodes)$. The convenient number of SNs should be lower than n-2, where n is the number of given nodes (GN) to be connected. Such a limit has been heuristically proven [2]. The fitness of the individual is the length of the SMT built using the GN, which are fixed, and the SN codified in the chromosome. The length of the tree depends on the positions of the SNs as specified in the chromosome.

2.1 The Initial Population

The first thing to do is to define a searching area or space in which the new SNs give good solutions. For instance, a node situated far from the GNs won't ever give an acceptable solution, e.g., a shorter tree. The first approach is to create the new nodes inside the polygon described by the GNs. The nodes are supposed to be in a plane, e.g., the GA works in two dimensions; however, other authors have tried to resolve this problem for three dimensions [9]. For simplicity, a rectangular polygon has been selected as a valid approximation. A minimum x and y, as well as a maximum x and y, are extracted from the GNs. The absolute values of the SNs coordinates are within these numbers. To form the chromosome, different coordinates are randomly chosen in accordance with a uniform probability density function. Therefore, a coordinate takes a value among 0 and 1 with equal probability and is the result of the normalisation of a given range in x and y to the [0,1] interval. Initially every chromosome is randomly created with the same length which, as pointed out previously, is equal to twice the maximum number of SNs ($l = 2xSN = 2x(GN - 2)$).

2.2 Building the Tree. The Fitness Function

Once the initial population has been formed, we need to build the trees represented by each of the individuals, taking account the GNs, and to calculate their lengths, which is adopted as a fitness measurement. The first step to build a minimum distance tree is to calculate the Euclidean distances between any two nodes, and link those which are nearest.

Connectivity and Distance Matrices. The calculated Euclidean distances are kept in a table called *distance matrix* that will be an essential part for the latter building of the minimal trees. The next step is to create the *connectivity matrix*. This matrix has a format quite similar to the previous one: a number '1' represents the existence of a link between two nodes, while a number '0'

means that there is no connection between the nodes. At the beginning all cells in this matrix hold a '0'. The first links are formed as follows: each node is linked to the one that, in accordance with the distance matrix, is nearest. When this process is ended each node is linked to any other node, though it doesn't mean that a connected tree has been formed. When there are two or more equal distances (and therefore, two or more possible links for a node), only one of them (randomly chosen) is formed. It is important to guarantee total connectivity: from a particular node any other node must be reached within a finite number of hops. For that purpose, an iterative process for testing the connectivity of each node has been developed. The main goal of this algorithm is to form as many links as needed to reach total connectivity, but assuring that the resulting length of the tree must be as short as possible. This algorithm can be briefly explained: it makes use of three elements:

- *Explorables:* it is a vector that contains as many elements as nodes the net has. These elements are '1' if somehow this node is accessible and '0' otherwise.
- *Explored:* it is a subset of the Explorables vector. These elements are '1' if the links of the node have been already explored, '0' otherwise.
- *Reachable:* is a matrix with the dimensions of the connectivity matrix that represents the reachable nodes for each particular node.

Once this matrix is formed, we test if all its elements are '1'. If this is the case, there is total connectivity. If it is not so, a link must be established, and it will be the shortest among all the possible ones according to the distance matrix. Then the Reachable matrix must be updated and so on until there is total connectivity.

The *fitness* of an individual is the sum of the lengths of all its links, since our goal was to get a Steiner minimal tree (SMT). The fitness is calculated combining the information stored in both the distance and the connectivity matrices.

Selection is carried through a tournament process. Four types of variation operators were considered: gluing, mutation, slicing and crossover. The gluing operation allows to concatenate a new SN to an existing chromosome. The concatenation takes place with a probability called P_g. The new SN is randomly selected within the searching space.

Mutation is a tool for the exploration of the neighbourhood of an individual. Each of the coordinates may be mutated with a probability P_m, therefore some coordinates may mutate inside a chromosome simultaneously. A gaussian mutation has been chosen, so the value of any coordinate (let's say the x coordinate) would be incremented/decremented an amount Δ x function of the distance to the mean of the gaussian curve. The mean variance settles the excursion range for mutations which is obtained linearly transforming an uniform distribution into a gaussian one,$x = gauss(0, \sigma)$

The process of dividing a chromosome by a random point is called slicing. It allows for the progressive shortening of the chromosomes, thus allowing trees with less SNs to go under selection to be compared with longer ones. This process

is also useful when trying to avoid local minima. A chromosome is sliced with a probability P_s. The result of the process are two shorter chromosomes, though only the owner of the best fitness will be in the next generation.

With a certain probability, P_c, a crossover process is performed on a chromosome. Similarly to the two previous processes, crossover is a useful tool when trying to avoid local minima. The crossover involves two chromosomes. The first one if the winner of the tournament and the second one is randomly selected from the subset of competitors initially chosen for the same tournament.

3 Results

The algorithm was executed and tested for a number of different input values; a set of typical values that gave good results was: 110 individuals, 50 generations, tournament size of 20, gluing, mutation, slicing and crossover probabilities of 0.2, 0.7, 0.05 and 0.05 repectively; σ was set to a thirtieth of the co-ordinate range.

The test problem on which we validated our algorithm is spanning tree for a $4x4$ Grid 16 GN problem (see Fig. 1 which also includes the graphical result of our algorithm). We also generated a solution to a randomly placed 100 node problem. Our results are presented in Table 1.

As far as the 16 GN problem is concerned, an improvement greater than 7% over the MST solution is a promising result when compared with the SMT optimal solution (obtained using an Annealing Algorithm which is effective only for small problems [6]); both solutions (the optimal and ours) introduce the same number of Steiner nodes, 10. Other authors have analysed the same problem proposing GAs with a similar number of function evaluations and a different scheme, getting improvements which almost reach 5% [7]. The main differences between the algorithm proposed in [7] and ours is in the crossover (they select two points in the chromosome randomly and then swap the information between the two points), in the mutation (which is a combination of gluing and slicing), and in the selection process. For the 100 node problem a direct comparison of our results (both MST and GA-based) with the optimal solution from [6] is not possible; the randomness of the distributions and the limited number of function evaluations in our algorithm (110x50)(same number as in the 16 node problem) are the reasons. Nevertheless, and for several random runs, we obtained improvements of around 1% over the MST solution(compared with 3% from [6]); the ratio of extra Steiner nodes versus given nodes (100) is small compared with the 16 node problem, meaning that the algorithm did not fully exploit the strength of adding extra Steiner nodes.

The sensitivity of the algorithm to changes in the inputs parameters (as P_m or P_g) is not high, which means that the search space is quite noisy. Mutation has been proven to be quite useful in such a noisy space because it allows to perform small movements from the individuals, looking for better fitnesses. Finally, the population size is a compromise solution because a higher number of individuals

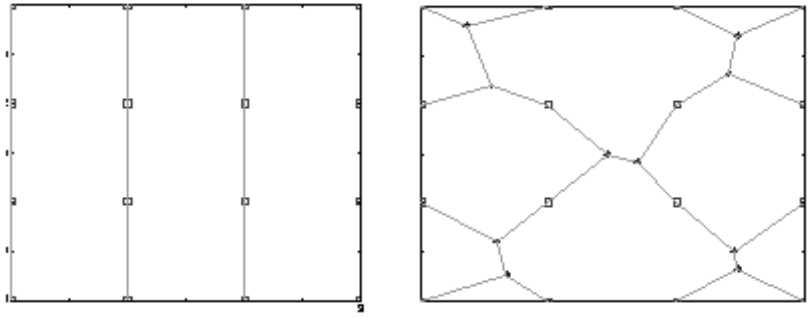


Fig. 1. Steiner Tree for a 4x4 Grid GNs before and after running the GA; square represent the fixed nodes, the remaining intersections are Steiner nodes

would allow a more accurate exploration of the space, meanwhile it also would significantly increase the execution time.

A very important step in the algorithm is the construction of the trees, which is the slowest of all the processes that compounds the algorithm. It is needed not only to calculate the Euclidean distances between any two nodes and link the nodes according with these measurements, but also to test that there is total connectivity to every node. This testing means a considerable computational effort. In order to speed up the algorithm, heuristics have to be applied, reducing the computing time. In our case, firstly we used the fact that the number of possible Steiner points is limited to n-2 for n GNs [2]; secondly, we exploited the fact that in a SMT the number of edges connected to a Steiner point , called its degree, should be three. A third particularity of these trees, the fact that the angle between the edges is 120 degrees has not been used for simplicity.

The selection operator, in this case a tournament process, is random enough to allow not such a good solutions to be selected; eventually it might be the case that these "bad" solutions may reach the best fitness of all ought to the slicing, gluing, crossover or mutation operators.

Table 1. Distance and percentage below the MST solution for the 16 GN problem

	16 Node problem	
MST Solution	3.00000	100%
SMT Optimal Solution	2.73205	91.0%
GA Solution	2.77412	92.5%

4 Conclusions and Future Work

The main goal of this paper has been was build a GA which would help solving an interesting problem in telecommunications such as the design of a backbone net. Our results, compared with the optimal solution and with those got by other authors, are acceptable.

An improvement to the algorithm would be to introduce a heating and cooling scheme for the mutation, that is to say, to change the variance of the Gaussian function according with certain parameters related with the algorithm evolution, and improving the exploration and exploitation of the search space. Work on this sort of mutation scheme has already been carried through [8]. Another improvement would be to take into consideration the fact that the angles between nodes should be 120 grades.

It is also our intention for the future that the fitness function should consider relevant parameters related to a particular problem, e.g., in a telecommunications net it would be important to observe what links support a greater load, what routers do specific functions, QoS, and preferred telecommunications companies amongst others.

References

1. S.L. Martins, C.C. Ribeiro, and M.C. Souza: A parallel GRASP for the Steiner problem in graphs. In A. Ferreira and J. Rolim, editors, Proceedings of IRREGULAR'98 – 5th International Symposium on Solving Irregularly Structured Problems in Parallel, Lecture Notes in Computer Science, Springer-Verlag, 1457 (1998) pp. 285–297.
2. E.N. Gilbert, H.O. Pollack: Steiner Minimal Trees. SIAM J.Appl.Math. Vol 16, No.1, 1986, pp.1–29.
3. B.A. Julstrom: A Genetic Algorithm for Rectilinear Steiner Problem. Fhith International Conference on Genetic Algorithms, 1993, pages 474–480.
4. J. Hesser, R. Manner and O. Stucky: Optimization of Steiner Trees Using Genetic Algorithms. Thrid International Conference on Genetic Algorithms, George Mason University, 1989, pages 231–237.
5. F.C. Harris Jr: Parallel Computation of Steiner Minimal Trees. PhD thesis, Clemson, University Clemson, SC 29634, May 1994.
6. F.C. Harris Jr: A stochastic optimization algorithm for Steiner minimal trees. Congr.Number., 105:54–64, 1994
7. J. Jones, F.C. Harris Jr: A Genetic Algorithm for the Steiner Minimal Tree Problem, Proc. ISCA's Int. Conf. on Intelligent Systems (IS '96) Reno, NV, June 19–21, 1996.
8. J.L. Fernández-Villacañas, S. Amin: Simulated Jumping in Genetic Algorithms for a set of test functions. Proceedings of the IASTED International Conference on Intelligent Information Systems, Grand Bahama Island, December, 1997.
9. M. Brazil, D.H. Lee, J.H. Rubinstein, D.A. Thomas, J.F. Weng and N.C. Wormald: Network Optimisation of Underground Mine Design. AusIMM Proceedings Volume 305, No 1, 2000.
10. H. Esbensen: Computing Near-Optimal Solutions to the Steiner Problem in a Graph Using a Genetic Algorithm. Networks 26 (1995), 173–185

Speaking Autonomous Intelligent Devices

Robert J. Ross, Bryan McEleney, Robert Kelly, Tarek Abu-Amer, Michael Walsh, Julie Carson-Berndsen, and Gregory M.P. O'Hare

Department of Computer Science, University College Dublin, Ireland
{robert.ross,bryan.mceleney,robert.kelly,tarek.abuamer,Michael.J.Walsh, Julie.Berndsen,Gregory.OHare}@ucd.ie

Abstract. The development of speech tools suitable for use in real world environments requires collaboration between computational linguistics and new implementation fields e.g. robotics, and the incorporation of new AI techniques to improve overall system performance. In this paper we present the core development concepts of SAID (Speaking Autonomous Intelligent Devices). The work presented centres around four key strands of research, namely the recasting of the *Time Map* model as a Multi-Agent System (MAS), the development of a MAS based audio feature extraction system, the deployment of a BDI based dialog agent and the design of a MAS based social robot architecture.

1 Introduction

This work builds on two hitherto separate research streams, computational linguistics/speech technology and MAS with the aim of developing a novel MAS architecture for use in speech recognition and synthesis. This work is motivated by a desire to support inter-device and human-device communication culminating in Speaking Autonomous Intelligent Devices. One aspect of this research is to recast an existing Time Map model in terms of a collection of interacting autonomous agents. The agents employed for this task are rich intentional agents governed by the Belief Desire Intention (BDI) [1] framework. The second aspect of this research is to link speech recognition and synthesis agents with higher level dialog and modelling agents responsible for planning and interpretation of utterances in the context of inter-device and human-device interactions. A MAS based social robot control architecture has been developed to run on a family of Nomad Scout II robots for this purpose.

In Sect. 2 we present background to the key research ideas behind our work, before moving on to Sect. 3 to give details of the development topics currently being undertaken as part of SAID.

2 Background Information

Due to the collaborative nature of SAID, there is a large quantity of background research upon which the project is built. In this section we introduce two of the main blocks of this research.

M. O'Neill et al. (Eds.): AICS 2002, LNAI 2464, pp. 240–245, 2002.

2.1 Autonomous Agents and Multi-agent Systems

Rather than being a passive recipient of actions by other entities, an agent is an active, independent, originator of action. Minimal qualities of an agent typically include autonomy, situatedness, and pro-active behaviour. The basic notion of an agent may then be built upon through the addition of qualities such as social ability, reactivity, intentionality etc. to produce agents which are suited to specific task domains.

Just as there are no clear advantages to the use of an object oriented design in a system which contains only one object, there are no real advantages to agent oriented design in a system with only one agent. Furthermore the advantages of agent design become most clear in the context of a system consisting of a number of agents. Such MAS have advantages of faster problem solving due to parallelism, system robustness, open system design, distribution of data or control, and easier legacy system integration. MAS is a bottom up approach to Distributed AI, which produces individual agents which are good at solving particular problems or aspects of problems.

2.2 The *Time Map* Model of Speech Recognition

The *Time Map* model is a computational linguistic model for speech recognition using declarative descriptions of phonological well-formedness constraints to interpret multilinear representations of speech utterances. The stages of recognition in this model are explained below and illustrated in Fig. 1. For a more detailed description of the *Time Map* model see [2] [3].

The first stage of recognition is to extract a set of features from the signal. The features can then be interpreted as a multilinear representation where each feature appears on one of a number of tiers. The well-formedness constraints are specified as a phonotactic automaton, which is a network representation of the legal sound combinations for the syllables of a language.

As the overlap relations in the multilinear representation are examined, they are tested against successive sets of constraints defined on the arcs of the automaton. Each arc of the automaton has a set of associated feature overlap constraints, each constraint having a numeric ranking. If the sum of the ranks for satisfied constraints on an arc equals or exceeds an associated threshold, that arc is traversed. If a final state is reached, a well-formed syllable has been found which is then passed to a lexicon which distinguishes between actual syllables appearing in the corpus and potential, i.e. phonologically well-formed, syllables of the language. Thus, it is possible to recognise new words not appearing in the corpus or even in the lexicon of the language.

3 SAID Project Development Topics

The SAID project is currently running on a number of key work paths, each of which has the aim of improving speech synthesis and recognition techniques in the mobile agent domain. In this section we briefly describe these paths.

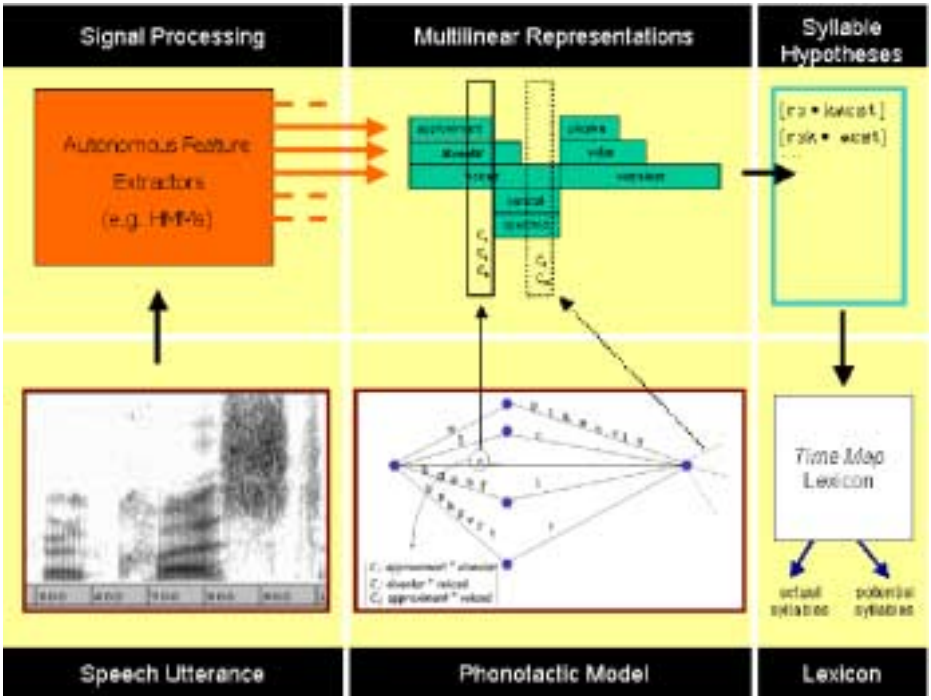


Fig. 1. The *Time Map* Model Architecture

3.1 Multi-agent *Time Map* Recognition

A key objective in realising SAID is to deploy MAS in the delivery of speech based systems. To achieve this it is necessary to recast the *Time Map* model in a MAS form.

A number of agents are required in order to accurately model the *Time Map* recognition process as illustrated in Fig. 2. Note that the arrows in Fig. 2 represent communication between agents. This communication ensures no duplication of work takes place. Furthermore, the system achieves robustness through constraint relaxation and output extrapolation (see [3]).

The agents that have been identified for the recognition task are

- *Syllable Agent* capable of activating a *Feature Extraction Agent* for each linguistic feature sought. A Syllable Agent is also capable of activating a number of *Chart Agents* and analysing any results communicated by them in an attempt to resolve the syllable recognition problem.
- *Feature Extraction Agent* capable of extracting a specific linguistic feature from the speech signal.
- *Chart Agent* capable of activating *Transition Agents* in order to plot a course through the phonotactic automaton.

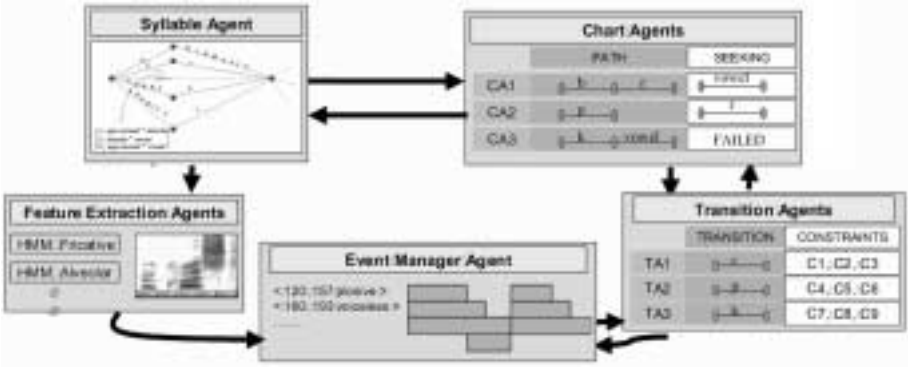


Fig. 2. Agent Architecture for *Time Map* recognition

- *Transition Agent* capable of determining if overlap relation constraints for a given transition are satisfied and communicating results to their governing Chart Agent.
- *Event Manager Agent* capable of acting as an intermediary between Feature Extraction Agents and Transition Agents.

This section has described multi-agent *Time Map* recognition at a high level, however more detail of the feature extraction process is required and is described in the next section.

3.2 Multi-agent Based Feature Extraction

Two distinctive approaches have been traditionally adopted for feature extraction, namely a knowledge-based approach and a data-driven approach. While the knowledge based approach applies certain constraints (defined in terms of energy levels, duration, spectrum shapes, formants transition, zero crossing rates etc.) for extracting features from the speech signal, the data driven approach relies on the prior learning of statistical systems e.g. Hidden Markov Models (HMMs) or Artificial Neural Networks (ANNs). In our design of the feature extractor we seek to combine the benefits of both approaches and cast them in a MAS. The feature extraction module is comprised of three consecutive layers that can be looked on as three agents cooperating to perform the task with the highest possible accuracy. Moving from the first towards the third layer, the features extracted increase in their level of granularity. This is briefly explained below:

- First Layer: a knowledge-based layer that applies signal processing methods to extract features of fine granularity from the speech signal e.g. speech/signal detection, close/release in plosives.
- Second Layer: uses a multi-linear system of HMMs to extract articulatory-acoustic features e.g. manner, place, voicing, height and rounding.
- Third Layer: uses multilayer perception (MLP) neural networks to adjust the coarsest level of features.

This is a novel design and it is hoped that it will outperform previous designs in this area. *Time Map* speech recognition can provide input text to any device, but for that device to be intelligent this input must be interpreted appropriately. In the next section we propose a cognitive dialog agent for this task.

3.3 Cognitive Dialog Agents

Most natural language dialogue systems have been rather constrained in the way a user query is elucidated. Some use a simple interview technique, where the system has full initiative, with the purpose of filling slots in a frame. Others allow the user to state a single question, which is interpreted, submitted to a knowledge base, and the results placed in a template response. We are interested in a richer dialogue pattern, where the system must understand the propositional content and speech act type of the user's utterances and infer their motivation. The system's utterances and actions should reflect a need to clarify the user's goals, and to formulate appropriate actions and utterances to satisfy those goals. For this we assume that the user is cooperative and has a set of goals in mind to which his utterances correspond.

A dialogue agent is proposed that will carry on a spoken dialogue with a user in the mobile agent domain. The dialog agent should:

- represent knowledge of a domain in a formalism suitable for retrieval and inference. Modal first order predicate calculus is suggested for this purpose, being amenable to inference and representation within the belief component of a BDI framework, and to integration with a relational database.
- participate in an appropriate dialog game with the user. At the present time, dialogue consists of a sequence of question-answer pairs. In future we expect the system to handle anaphoric reference and ellipsis, and engage in more elaborate dialogue structures, such as negotiation or clarification.
- maintain a representation of the user's goals and plans. The agent incorporates a planning system to generate appropriate speech and possibly domain actions [4] that further the user's goals, and also a plan recognition component to infer the user's plans from their speech actions [5].

The dialog agent proposed above could be used in any number of applications. In particular it could act as one high level component functioning in a robot control architecture such as that described below.

3.4 SRA++

The Social Robotic Architecture (SRA) [6] was a robot control architecture developed to allow a group of robots to interact as high level social entities. SRA++ aims to build upon the work on the SRA by taking full advantage of a truly MAS based design.

SRA++ is a hybrid MAS based robot control architecture, using reactive style agents to provide basic reactive behaviours, while using intentional agents

to provide deliberative control, planning and modelling. SRA++ is effectively a social robot architecture due to the incorporation of agents that model and reason about other agents (both human and artificial) in the environment. A key difference between SRA++ and the SRA is that SRA++ allows for explicit social communication through a number of mediums.

Agent Factory [1] and its successor Agent Factory Lite are agent prototyping environments which allow for the creation and deployment of intentional agents. While the development of intentional agents for use within SRA++ is provided by Agent Factory, some modifications were required to enable reactive agents to be integrated alongside intentional agents.

As part of the development process all speech and user modelling agents discussed above will be tested within SRA++. In addition, tests involving the use of a visual system to prime the speech recognition system (based on the presence of articles in the robots environment) are being evaluated.

4 Conclusions and Future Work

In this paper the key strands of research constituting SAID were presented. Novel work includes the incorporation of MAS design in the development of feature extraction and computational linguistic tools. Other novel work includes the development of an agent-based dialog system and the design of a robot architecture with audio integration at a number of levels. Based on the design issues raised in this paper, future work will centre on furthering agent integration and deployment on mobile agents.

Acknowledgements. We gratefully acknowledge the support of Enterprise Ireland through grant No. IF/2001/02, SAID.

References

1. Collier, R.W.: Agent Factory: A Framework for hte Engineering of Agent Oriented Applications. PhD thesis, University College Dublin (2001)
2. Carson-Berndsen, J.: Finite state models, event logics and statistics in speech recognition. Philosophical Transactions of the Royal Society (2000)
3. Carson-Berndsen, J., Walsh, M.: Phonetic time maps: Defining constraints for multilinear speech processing. In Dommelen, V., Barry, eds.: Phonetic Knowledge in Speech Technology. Kluwer Academic Publishers (To Appear)
4. Austin, J.L.: How to do things with Words. Harvard University Press (1962)
5. Allen, J.F., Perrault, C.R.: Analyzing intention in utterances. In Grosz, K.S.J.B.J., Webber, B.L., eds.: Readings in Natural Language Processing. Morgan Kaufmann, Los Altos (1980) 441–458
6. Duffy, B.R.: The Social Robot. PhD thesis, University College Dublin (2000)

Author Index